Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка iOS-застосунку для on-device розпізнавання мовлення в реальному часі з використанням Core ML
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-24-1
Нікіта СТОРЧАЙ
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва освітньої програми)
Керівник проф. Ваган ТЕРЗІЯН
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Лариса ЧАЛА
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сторчай Нікіті Павловичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка iOS-застосунку для on-device розпізнавання мовлення в реальному часі з використанням Core ML

затверджена наказом університету від 6 квітня 20 26 р. № 269Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 19 травня 20 26 р.

3. Вихідні дані до роботи наукові статті та технічна документація з області автоматичного розпізнавання мовлення, on-device машинного навчання та обробки природної мови, документація Apple щодо Swift, SwiftUI, SwiftData, Core ML та Foundation Models, матеріали щодо архітектури та інтеграції Whisper-моделей у мобільне середовище, а також вихідні коди й приклади реалізації iOS-застосунків із локальною обробкою аудіоданих.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Теоретичні дослідження

3) Проектування системи

4) Програмна реалізація

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	06.04.2026	виконано
2	Визначення проблеми	08.04.2026	виконано
3	Аналіз конкурентних рішень	12.04.2026	виконано
4	Проектування компонентів системи	20.04.2026	виконано
5	Реалізація застосунку	26.04.2026	виконано
6	Оформлення пояснювальної записки	01.05.2026	виконано
7	Захист	19.05.2026	виконано

Дата видачі завдання 6 квітня 20 26 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Ваган ТЕРЗІЯН
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 127 с., 36 рис., 1 дод., 20 джерел.

АВТОМАТИЧНЕ РОЗПІЗНАВАННЯ МОВЛЕННЯ, МАШИННЕ НАВЧАННЯ, МОБІЛЬНИЙ ЗАСТОСУНОК, ОБРОБКА МОВЛЕННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, РЕАЛЬНИЙ ЧАС, ШТУЧНИЙ ІНТЕЛЕКТ, CORE ML, ON-DEVICE AI, SPEECH RECOGNITION, TRANSFORMER, WHISPER.

Об’єктом дослідження є процес автоматичного розпізнавання мовлення у мобільних програмних системах.

Предметом дослідження є засоби реалізації on-device моделй розпізнавання мовлення на платформі iOS із використанням Core ML.

Метою роботи є розробка мобільного застосунку для розпізнавання мовлення в реальному часі безпосередньо на пристрої користувача з забезпеченням високої точності та низької затримки.

Методами дослідження є аналіз сучасних підходів до автоматичного розпізнавання мовлення, використання глибоких нейронних мереж трансформерного типу, а також застосування інструментів машинного навчання та фреймворків платформи iOS.

У результаті роботи розроблено прототип iOS-застосунку, що реалізує on-device розпізнавання мовлення у реальному часі із використанням оптимізованої моделі машинного навчання та забезпечує ефективну обробку аудіосигналу без передачі даних на зовнішні сервери.

Отримані результати можуть бути використані при розробці мобільних застосунків для автоматичної транскрипції мовлення, голосового введення інформації та створення інтелектуальних інтерфейсів взаємодії з користувачем. Запропоноване рішення доцільно застосовувати у сферах освіти, бізнесу, медіа та персональної продуктивності.

ABSTRACT

Master’s thesis contains: 127 pp., 33 fig., 1 ann., 20 references.

ARTIFICIAL INTELLIGENCE, AUTOMATIC SPEECH RECOGNITION, CORE ML, MACHINE LEARNING, MOBILE APPLICATION, NATURAL LANGUAGE PROCESSING, ON-DEVICE AI, REAL TIME, SPEECH PROCESSING, SPEECH RECOGNITION, TRANSFORMER, WHISPER.

The object of the research is the process of automatic speech recognition in mobile software systems.

The subject of the research is the means of implementing on-device speech recognition models on the iOS platform using Core ML.

The purpose of the work is to develop a mobile application for real-time speech recognition directly on the user’s device while ensuring high accuracy and low latency.

The research methods include the analysis of modern approaches to automatic speech recognition, the use of deep neural networks of the transformer type, as well as the application of machine learning tools and iOS platform frameworks.

As a result of the work, a prototype of an iOS application was developed that implements real-time on-device speech recognition using an optimized machine learning model and provides efficient audio signal processing without transmitting data to external servers.

The obtained results can be used in the development of mobile applications for automatic speech transcription, voice input, and the creation of intelligent user interaction interfaces. The proposed solution is suitable for use in education, business, media, and personal productivity.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 11
Вступ 13
1 Аналіз предметної галузі та постановка задачі 15
1.1 Основи розпізнавання мовлення 15
1.1.1 Поняття автоматичного розпізнавання мовлення та його еволюція 16
1.1.2 Ключові етапи обробки аудіосигналу у системах ASR 18
1.1.3 Роль нейронних мереж у сучасних системах розпізнавання мовлення 19
1.2 Технології on-device розпізнавання мовлення 21
1.2.1 Переваги локальної обробки мовлення на телефоні 23
1.2.2 Вплив обмежених ресурсів мобільних платформ на моделі 23
1.2.3 Порівняння on-device та cloud-based підходів 24
1.3 Моделі та алгоритми для розпізнавання мовлення 26
1.3.1 Класичні підходи: HMM та GMM 27
1.3.2 Глибокі нейронні мережі у задачах ASR 28
1.3.3 Архітектура моделей типу encoder-decoder та трансформери .. 29
1.3.4 Особливості моделі Whisper для розпізнавання мовлення 31
1.4 Технологічне забезпечення iOS для роботи з мовленням 32
1.4.1 Можливості фреймворків AVFoundation та Speech 34
1.4.2 Використання Core ML для інтеграції моделей машинного навчання 35
1.4.3 Оптимізація моделей для роботи на мобільних пристроях 35
1.5 Аналіз сценаріїв використання мобільного застосунку 36
1.5.1 Основні сценарії взаємодії користувача із системою 37
1.5.2 Вимоги до обробки мовлення в реальному часі 38
1.6 Формування вимог до програмної системи 38
1.6.1 Функціональні вимоги до застосунку 39

1.6.2 Нефункціональні вимоги: продуктивність, затримка, енергоефективність 39
1.6.3 Вимоги до конфіденційності та безпеки даних 40
1.7 Постановка задачі 40
2 Теоретичні дослідження 42
2.1 Теоретичні основи цифрового подання мовленнєвого сигналу 42
2.1.1 Фізична природа мовлення як акустичного сигналу 43
2.1.2 Дискретизація, квантування та частота семплювання аудіосигналу 44
2.1.3 Представлення мовлення у часовій та частотній областях 46
2.2 Методи виділення ознак з аудіосигналу 48
2.2.1 Спектральний аналіз мовленнєвого сигналу 49
2.2.2 Мел-спектрограма як вхідне представлення для нейронних моделей 50
2.2.3 Вплив шуму, пауз і довжини аудіофрагмента на якість розпізнавання 52
2.3 Архітектурні принципи сучасних моделей розпізнавання мовлення 54
2.3.1 Кодування аудіопослідовності у нейромережевих моделях 56
2.3.2 Декодування тексту за акустичними ознаками 57
2.3.3 Механізм уваги у задачах перетворення мовлення в текст 58
2.3.4 Трансформерна архітектура в системах ASR 59
2.4 Теоретичні особливості моделі Whisper 61
2.4.1 Загальна концепція моделі Whisper 62
2.4.2 Перетворення аудіосигналу у лог-мел-спектрограму 64
2.4.3 Encoder-decoder структура моделі Whisper 66
2.4.4 Багатомовність, токенізація та формування текстового результату 67
2.5 Потокове розпізнавання мовлення в реальному часі 69
2.5.1 Принцип обробки аудіопотоку фрагментами 69
2.5.2 Баланс між затримкою розпізнавання та точністю 70

2.5.3 Проблема контексту під час інкрементального формування тексту 70
2.5.4 Стабілізація проміжних результатів транскрипції 71
2.6 Теоретичні аспекти оптимізації моделей для мобільних пристроїв .. 71
2.6.1 Обмеження обчислювальних ресурсів iOS-пристроїв 73
2.6.2 Квантування та зменшення розміру моделей машинного навчання 73
2.6.3 Використання апаратного прискорення для виконання нейронних моделей 74
2.6.4 Вплив оптимізації на швидкодію та точність 74
2.7 Інтеграція моделей машинного навчання у Core ML 75
2.7.1 Формат моделей Core ML та особливості їх виконання 77
2.7.2 Використання Neural Engine, GPU та CPU для локального інференсу 79
2.7.3 Обмеження та переваги Core ML у задачах on-device ASR 80
2.7.4 Особливості використання Whisper-моделі в iOS 81
3 Проектування системи 82
3.1 Формалізація вимог до мобільного застосунку 82
3.1.1 Функціональні вимоги до системи розпізнавання мовлення 82
3.1.2 Нефункціональні вимоги до продуктивності та затримки обробки 83
3.1.3 Вимоги до конфіденційності та on-device обробки даних 84
3.1.4 Вимоги до користувацького інтерфейсу 84
3.2 Проєктування функціональної моделі системи 85
3.2.1 Модуль захоплення та обробки аудіосигналу 86
3.2.2 Модуль розпізнавання мовлення на основі Whisper і Core ML 86
3.2.3 Модуль формування та збереження текстових нотаток 87
3.2.4 Модуль AI-обробки та трансформації транскрипцій 87
3.2.5 Модуль керування локальними даними 87
3.3 Моделювання сценаріїв взаємодії користувача із системою 88

3.3.1 Основні користувацькі потоки мобільного застосунку 88
3.3.2 Сценарій створення голосової нотатки в реальному часі 89
3.3.3 Сценарій перегляду та редагування транскрипції 90
3.3.4 Сценарій AI-перетворення текстових нотаток 91
3.3.5 Схема використання мобільного застосунку 92
3.4 Проєктування архітектури мобільного застосунку 94
3.4.1 Обґрунтування вибору технологічного стеку 95
3.4.2 Архітектурний підхід та організація компонентів системи 96
3.4.3 Організація on-device ML pipeline для реального часу 96
3.4.4 Інтеграція Whisper-моделі через Core ML 97
3.4.5 Використання Apple Foundation Models для обробки тексту 97
3.5 Проєктування інтерфейсу мобільного застосунку 97
3.5.1 Концепція UX/UI для системи голосових нотаток 99
3.5.2 Проєктування екрана запису та live-транскрипції 99
3.5.3 Проєктування бібліотеки нотаток та екрана перегляду 99
3.5.4 Візуалізація станів запису та обробки аудіо 100
4 Програмна реалізація 101
4.1 Вибір технологій та засобів реалізації 101
4.1.1 Використання SwiftUI для створення iOS-застосунку 101
4.1.2 Застосування Core ML для розпізнавання мовлення 102
4.1.3 Інтеграція Whisper-моделі у середовище iOS 102
4.1.4 Використання SwiftData для локального збереження транскрипцій 103
4.1.5 Використання Apple Foundation Models для інтелектуальної обробки нотаток 103
4.2 Реалізація основної функціональності застосунку 104
4.2.1 Реалізація запису та обробки мовлення 105
4.2.2 Реалізація транскрибування мовлення у реальному часі 107
4.2.3 Реалізація збереження та перегляду нотаток 110
4.2.4 Реалізація інтелектуальної обробки тексту 112

4.3 Реалізація інтерфейсу користувача 114
Висновки 123
Перелік джерел посилання 125
Додаток А Відомість кваліфікаційної роботи 127

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект, напрям комп'ютерних наук, що охоплює методи створення систем, здатних виконувати інтелектуальні задачі;

Apple Foundation Models – набір локальних генеративних моделей Apple, що використовується для інтелектуальної обробки, переформатування та покращення текстових нотаток;

Apple Intelligence – система інтелектуальних функцій Apple, що забезпечує локальну AI-обробку даних у підтримуваних пристроях;

ASR – Automatic Speech Recognition – автоматичне розпізнавання мовлення, задача перетворення аудіосигналу з мовленням у текстовий формат;

AVAudioEngine – програмний інструмент Apple для роботи з аудіопотоками, записом, обробкою та маршрутизацією аудіосигналу в iOS-застосунках;

Core ML – фреймворк Apple для інтеграції та виконання моделей машинного навчання безпосередньо на пристроях екосистеми Apple;

CPU – Central Processing Unit – центральний процесор, універсальний обчислювальний модуль пристрою;

Foundation Models – генеративні моделі Apple, що використовуються для локальної інтелектуальної обробки тексту;

GPU – Graphics Processing Unit – графічний процесор, який може використовуватися для прискорення паралельних обчислень під час виконання моделей машинного навчання;

HIG – Human Interface Guidelines – рекомендації Apple щодо проєктування інтерфейсів користувача для застосунків екосистеми Apple;

iOS – мобільна операційна система Apple, що використовується на пристроях iPhone;

ML – Machine Learning – машинне навчання, підхід штучного інтелекту, що передбачає навчання моделей на основі даних;

MV – Model-View – архітектурний підхід, що передбачає розділення представлення інтерфейсу та моделі стану застосунку;

Neural Engine – спеціалізований апаратний модуль Apple для прискореного виконання операцій машинного навчання;

NLP – Natural Language Processing – обробка природної мови, напрям штучного інтелекту, пов’язаний з аналізом, генерацією та перетворенням текстових даних;

On-device – підхід до обробки даних, за якого обчислення виконуються локально на пристрої без передачі інформації на зовнішні сервери;

PCM – Pulse-Code Modulation – імпульсно-кодова модуляція, цифрове представлення аудіосигналу, що використовується для подальшої обробки мовлення;

Pipeline – послідовність етапів обробки даних, у межах якої результат одного етапу передається на наступний;

Prompt – текстова інструкція для генеративної моделі, яка визначає бажаний формат, стиль або зміст результату;

SwiftUI – декларативний фреймворк Apple для створення користувацьких інтерфейсів;

UI – User Interface – користувацький інтерфейс, сукупність візуальних і функціональних елементів взаємодії користувача із застосунком;

UX – User Experience – користувацький досвід, що характеризує зручність, логіку та якість взаємодії користувача із системою;

Whisper – модель автоматичного розпізнавання мовлення, що використовується для перетворення аудіосигналу у текстову транскрипцію.

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням ролі інтелектуальних систем, здатних обробляти природну мову людини. Одним із ключових напрямів у цій сфері є автоматичне розпізнавання мовлення, яке дозволяє перетворювати аудіосигнал у текстову форму та забезпечувати природну взаємодію користувача з цифровими пристроями. Такі технології активно впроваджуються у мобільні застосунки, голосові асистенти, системи автоматизації, а також у спеціалізовані сервіси для обробки аудіоконтенту. Розвиток алгоритмів машинного навчання, зокрема глибоких нейронних мереж, значно підвищив точність розпізнавання мовлення, що сприяло їх широкому використанню у різних галузях.

Незважаючи на значні досягнення у цій сфері, існує низка проблем, пов’язаних із забезпеченням швидкодії, стабільності та конфіденційності під час обробки мовлення. Більшість сучасних рішень передбачає використання хмарних сервісів, що потребує постійного підключення до мережі Інтернет та супроводжується передачею аудіоданих на віддалені сервери. Це, у свою чергу, створює ризики витоку персональної інформації та обмежує можливість використання таких систем у середовищах із підвищеними вимогами до безпеки. Крім того, залежність від мережі може негативно впливати на швидкість обробки даних та знижувати загальну ефективність роботи застосунку.

У відповідь на зазначені виклики активно розвивається підхід до виконання обчислень безпосередньо на пристрої користувача, відомий як on-device машинне навчання. Завдяки сучасним апаратним можливостям мобільних пристроїв та спеціалізованим фреймворкам, таким як Core ML, стало можливим ефективне виконання складних моделей без необхідності звернення до зовнішніх серверів. Це дозволяє не лише зменшити затримки

при обробці мовлення, але й суттєво підвищити рівень конфіденційності даних, що є критично важливим для сучасних користувачів.

Актуальність даної роботи зумовлена зростаючим попитом на мобільні застосунки, що забезпечують швидке, точне та безпечне розпізнавання мовлення. Особливої ваги набуває необхідність створення рішень, які можуть функціонувати незалежно від мережевого підключення та гарантувати захист персональних даних користувача. Розробка таких систем є важливим кроком у напрямку підвищення якості взаємодії людини з мобільними пристроями та розширення функціональних можливостей сучасних застосунків.

Очікуваним результатом виконання роботи є створення програмного продукту, який дозволяє здійснювати автоматичну транскрипцію мовлення без використання зовнішніх серверів. Такий підхід забезпечує високу швидкість обробки, мінімальні затримки та підвищений рівень конфіденційності. Крім того, реалізована система може слугувати основою для подальшого розширення функціональності, зокрема інтеграції додаткових алгоритмів аналізу тексту або підтримки багатомовного розпізнавання.

Розроблений застосунок може знайти широке застосування у різних сферах, включаючи освіту, бізнес, журналістику та особисту продуктивність. Він може використовуватися для створення текстових нотаток, протоколювання зустрічей, автоматичного створення транскрипцій аудіозаписів, а також як допоміжний інструмент для людей із обмеженими можливостями. Таким чином, результати даної роботи мають як наукову, так і практичну цінність, сприяючи подальшому розвитку технологій обробки мовлення у мобільному середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Основи розпізнавання мовлення

Автоматичне розпізнавання мовлення є одним із ключових напрямів розвитку сучасних інформаційних технологій, що забезпечує природну взаємодію користувача з програмними системами. Основною метою таких систем є перетворення акустичного сигналу у текстове представлення з подальшою можливістю його використання у різних прикладних задачах. Типова структура системи обробки мовлення включає декілька послідовних етапів, серед яких захоплення аудіосигналу, його попередня обробка, виділення ознак та застосування моделі машинного навчання. На першому етапі, відповідно до наведеної структури (рисунок 1.1), здійснюється автоматичне розпізнавання мовлення, яке передбачає перетворення звукового сигналу у текстову форму.

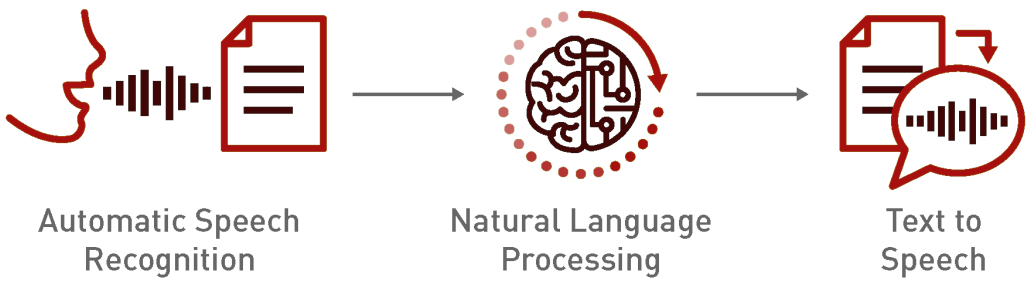


Рисунок 1.1 – Схема обробки мовлення у сучасних інтелектуальних системах

Для цього використовуються методи цифрової обробки сигналів, що дозволяють виділяти релевантні характеристики мовлення, такі як спектральні ознаки, частотні компоненти та часові залежності. Сучасні підходи базуються на використанні глибоких нейронних мереж, які здатні ефективно моделювати складні закономірності мовлення та забезпечувати високу точність розпізнавання навіть у присутності шумів або змінних умов запису.

Наступним етапом є обробка природної мови, яка також представлена у структурі системи (рисунок 1.1) та відповідає за інтерпретацію отриманого тексту. На цьому рівні виконуються задачі нормалізації тексту, визначення його структури та семантичного змісту. Хоча у межах даної роботи основний акцент зроблено саме на процесі розпізнавання мовлення, інтеграція з модулями обробки природної мови дозволяє розширити функціональність системи, зокрема реалізувати інтелектуальні сценарії взаємодії з користувачем.

Завершальним етапом, відповідно до загальної схеми (рисунок 1.1), може виступати синтез мовлення або збереження отриманого тексту для подальшого використання. У контексті розроблюваного мобільного застосунку основний акцент робиться на формуванні текстових нотаток у реальному часі без використання хмарних сервісів. Це зумовлює необхідність оптимізації алгоритмів розпізнавання для роботи безпосередньо на пристрої, що забезпечує підвищений рівень конфіденційності даних та незалежність від мережевого з’єднання.

1.1.1 Поняття автоматичного розпізнавання мовлення та його еволюція

Автоматичне розпізнавання мовлення являє собою технологію перетворення усного мовлення людини у текстову форму засобами обчислювальних систем. У загальному випадку така технологія поєднує

методи цифрової обробки сигналів, статистичного аналізу та машинного навчання для виявлення закономірностей у мовному потоці. Практичне значення автоматичного розпізнавання мовлення постійно зростає у зв'язку з поширенням голосових інтерфейсів, мобільних застосунків, систем диктування та сервісів транскрибування, для яких критично важливими є швидкість, точність та зручність взаємодії.

Типовий процес функціонування системи автоматичного розпізнавання мовлення включає послідовні етапи захоплення аудіосигналу, його попередньої обробки, виділення акустичних ознак, побудови акустичної та мовної моделі, а також декодування кінцевого результату у текстову форму (рисунок 1.2). Така структура відображає логіку переходу від безперервного звукового сигналу до дискретного текстового представлення. Саме узгоджена робота зазначених компонентів забезпечує можливість коректного розпізнавання слів, урахування мовного контексту та зменшення кількості помилок у фінальній транскрипції.

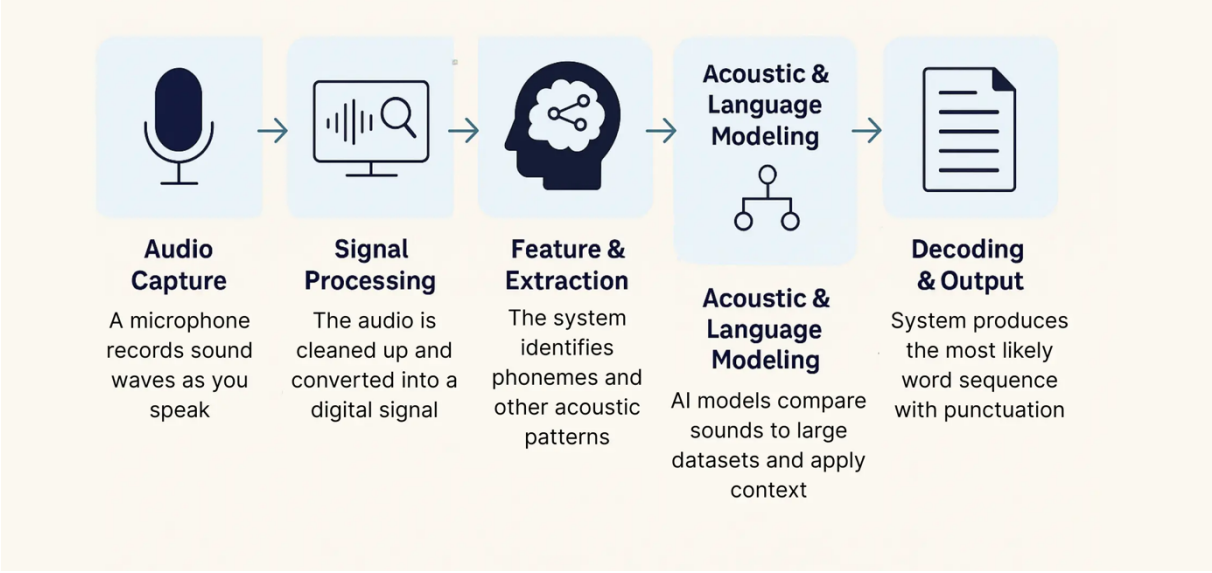


Рисунок 1.2 – Етапи автоматичного розпізнавання мовлення

Еволюція систем автоматичного розпізнавання мовлення пройшла декілька етапів. Перші підходи ґрунтувалися на шаблонному зіставленні,

що давало змогу розпізнавати лише обмежені набори слів у контрольованих умовах. У подальшому широкого застосування набули статистичні методи, зокрема моделі прихованих марковських процесів у поєднанні з гаусовими сумішами, які суттєво покращили якість розпізнавання безперервного мовлення. Сучасний етап розвитку пов'язаний із використанням глибоких нейронних мереж і трансформерних архітектур, що дозволили значно підвищити точність, адаптивність та стійкість систем до шумів і варіативності мовлення.

1.1.2 Ключові етапи обробки аудіосигналу у системах ASR

Обробка аудіосигналу є базовим етапом функціонування систем автоматичного розпізнавання мовлення, оскільки саме на цьому рівні відбувається перетворення аналогового звукового сигналу у форму, придатну для подальшого аналізу. Якість виконання цього етапу безпосередньо впливає на точність кінцевого результату розпізнавання. Основні стадії обробки аудіосигналу (рисунок 1.3) включають захоплення сигналу, попередню обробку, виділення ознак, побудову акустичної моделі, мовне моделювання та декодування результату.

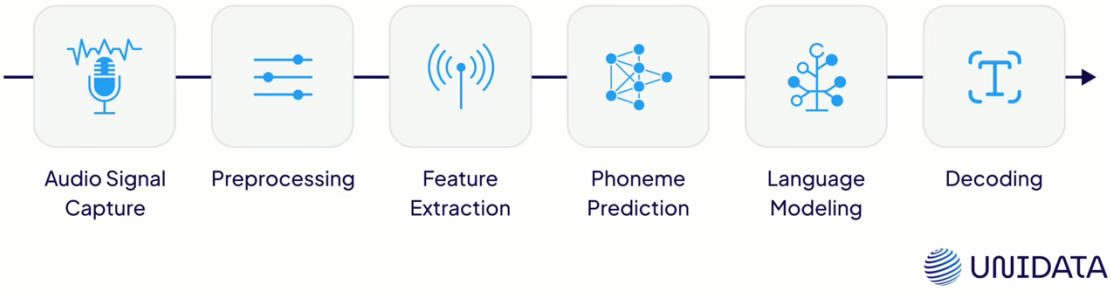


Рисунок 1.3 – Ключові етапи обробки аудіосигналу у системах автоматичного розпізнавання мовлення

Першим етапом, відповідно до наведеної структури (рисунок 1.3), є захоплення аудіосигналу з мікрофона, після чого сигнал піддається попередній обробці. На цьому етапі виконуються операції нормалізації, фільтрації шумів та сегментації сигналу, що дозволяє підвищити якість вхідних даних. Особливо важливим є усунення фонових шумів та стабілізація рівня сигналу, що є критичним для мобільних пристроїв, які працюють у різноманітних акустичних умовах.

Наступним кроком є виділення ознак, що також відображено у загальній схемі (рисунок 1.3). На цьому етапі із сирого аудіосигналу формуються компактні представлення, які зберігають ключову інформацію про мовлення. Найбільш поширеними є спектральні ознаки, зокрема мел-частотні кепстральні коефіцієнти, які дозволяють ефективно описувати частотні характеристики сигналу. Отримані ознаки використовуються як вхідні дані для подальших моделей машинного навчання.

Після цього здійснюється прогнозування фонем та застосування мовної моделі, що забезпечує врахування контексту та граматичних закономірностей мови (рисунок 1.3). Акустична модель встановлює відповідність між ознаками сигналу та елементарними мовними одиницями, тоді як мовна модель визначає найбільш ймовірні послідовності слів. Завершальним етапом є декодування, у процесі якого формується фінальний текстовий результат. Таким чином, ефективна організація кожного з етапів обробки аудіосигналу є критично важливою для забезпечення точності та швидкодії систем розпізнавання мовлення, особливо у задачах реального часу.

1.1.3 Роль нейронних мереж у сучасних системах розпізнавання мовлення

Сучасні системи автоматичного розпізнавання мовлення значною мірою базуються на використанні нейронних мереж, які забезпечують

високий рівень точності та адаптивності до різних умов мовлення. На відміну від класичних статистичних підходів, нейронні мережі здатні автоматично виявляти складні нелінійні залежності між акустичними ознаками та мовними одиницями. Це дозволяє ефективно працювати з варіативністю вимови, шумами та різними мовними контекстами.

Нейронні мережі застосовуються (рисунок 1.4) на декількох етапах обробки мовлення, починаючи з покращення якості сигналу та закінчуючи безпосереднім розпізнаванням фонетичних і текстових одиниць. Після етапу попередньої обробки та виділення ознак, отримані дані передаються до акустичної моделі, реалізованої у вигляді глибокої нейронної мережі. Така модель виконує зіставлення між вхідними ознаками та ймовірнісним розподілом мовних одиниць, що дозволяє визначити найбільш ймовірну інтерпретацію мовлення. Важливою особливістю є можливість використання архітектур типу рекурентних нейронних мереж та трансформерів, які враховують часову залежність у мовному сигналі.

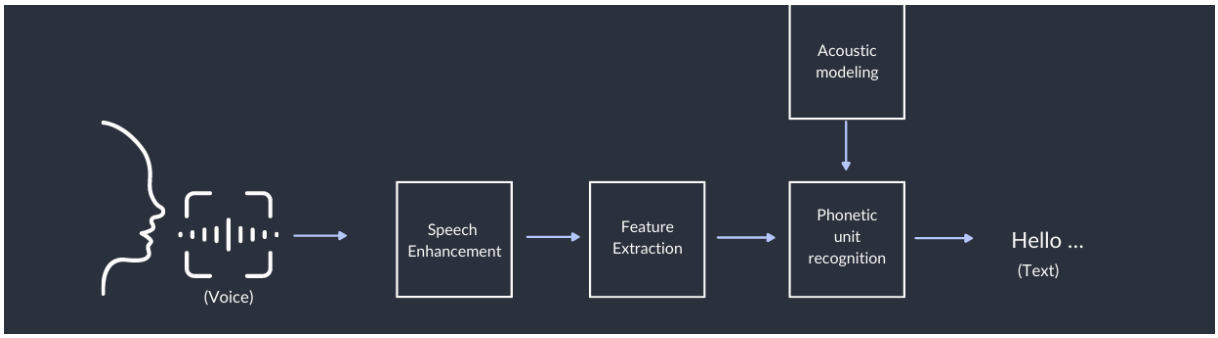


Рисунок 1.4 – Використання нейронних мереж у процесі розпізнавання мовлення

Окрім акустичного моделювання, нейронні мережі активно застосовуються і на етапі мовного моделювання (рисунок 1.4), де вони забезпечують врахування контексту та синтаксичних особливостей мови. Завдяки цьому підвищується якість кінцевого текстового результату, зменшується кількість помилок та покращується узгодженість слів у

реченні. У сучасних системах дедалі частіше використовуються наскрізні моделі, які об’єднують декілька етапів обробки у єдину нейронну архітектуру, що додатково спрощує процес розробки та підвищує ефективність.

У контексті мобільних застосунків особливу роль відіграє оптимізація нейронних мереж для виконання безпосередньо на пристрої. Це передбачає зменшення розміру моделей, використання квантизації та апаратного прискорення, зокрема через фреймворк Core ML. Завдяки цьому стає можливим реалізувати розпізнавання мовлення у реальному часі без використання хмарних сервісів, що забезпечує зниження затримок та підвищення рівня конфіденційності користувацьких даних.

1.2 Технології on-device розпізнавання мовлення

Технології on-device розпізнавання мовлення передбачають виконання всіх обчислювальних процесів безпосередньо на пристрої користувача без передачі аудіоданих до зовнішніх серверів. Такий підхід забезпечує підвищений рівень конфіденційності, зменшення затримок обробки та незалежність від мережевого з’єднання. У сучасних мобільних застосунках це особливо важливо для реалізації функцій реального часу, таких як безперервна транскрипція мовлення.

Архітектура on-device систем базується на взаємодії декількох ключових компонентів, серед яких центральне місце займають акустична та мовна моделі (рисунок 1.5). Акустична модель відповідає за перетворення аудіосигналу у послідовність базових мовних одиниць, тоді як мовна модель виконує уточнення результату з урахуванням контексту та статистичних закономірностей мови. У межах підходу SpeechAnalyzer ці компоненти інтегруються в єдиний конвеєр обробки, що дозволяє отримувати більш точні результати розпізнавання навіть у складних умовах.

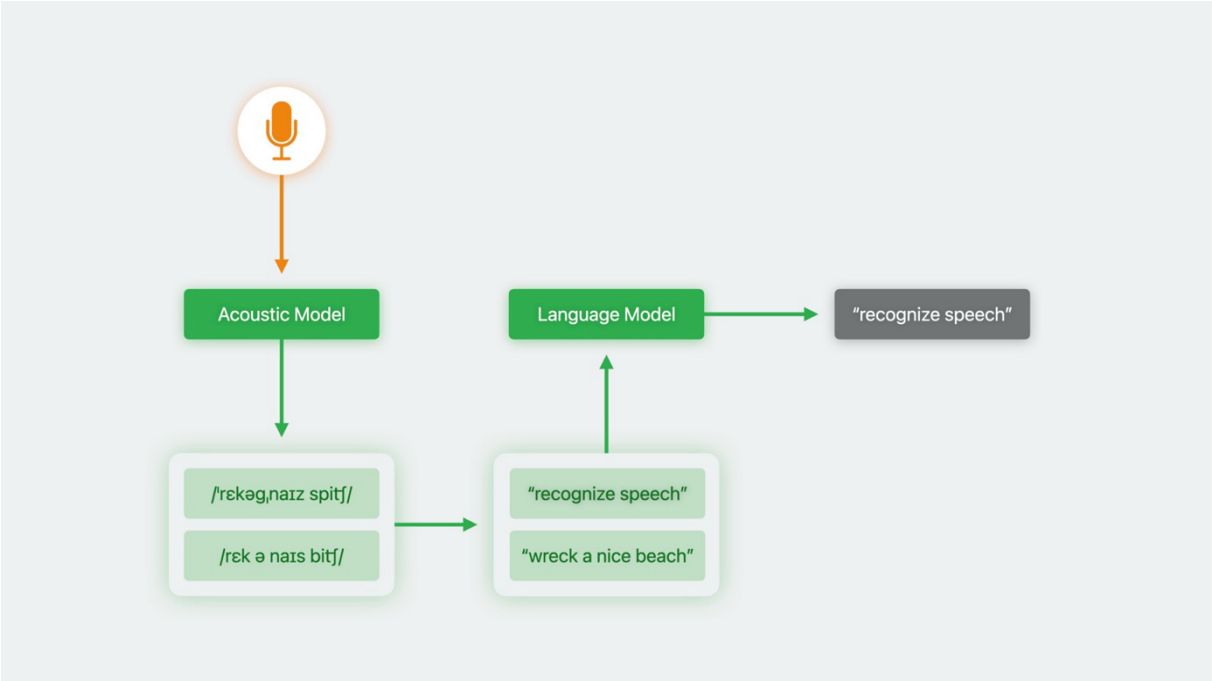


Рисунок 1.5 – SpeechAnalyzer для розпізнавання мовлення

Особливістю реалізації on-device розпізнавання мовлення є необхідність оптимізації моделей для роботи в умовах обмежених ресурсів мобільних пристроїв. Це включає зменшення обчислювальної складності, використання компактних архітектур нейронних мереж та застосування механізмів апаратного прискорення. У середовищі iOS важливу роль у цьому процесі відіграє використання Core ML, який забезпечує ефективне виконання моделей безпосередньо на пристрої, використовуючи доступні апаратні ресурси.

Крім того, застосування on-device підходу дозволяє реалізувати безперервну обробку мовлення у реальному часі без необхідності очікування відповіді від сервера. Це забезпечує більш природний користувацький досвід та відкриває можливості для створення інтерактивних голосових інтерфейсів. Таким чином, використання технологій локального розпізнавання мовлення є ключовим фактором у розробці сучасних мобільних застосунків.

1.2.1 Переваги локальної обробки мовлення на телефоні

Локальна обробка мовлення безпосередньо на мобільному пристрої забезпечує високий рівень конфіденційності користувацьких даних, оскільки аудіосигнал не передається на віддалені сервери. Це особливо важливо у випадках роботи з персональною інформацією, голосовими нотатками або приватними розмовами. Відсутність передачі даних мережею зменшує ризики перехоплення інформації та відповідає сучасним вимогам щодо захисту персональних даних.

Ще однією суттєвою перевагою є зменшення затримок обробки, що є критично важливим для систем реального часу. Оскільки всі обчислення виконуються локально, відсутня необхідність у передачі аудіо на сервер і очікуванні відповіді, що значно прискорює процес розпізнавання. Це дозволяє реалізувати безперервну транскрипцію мовлення з миттєвим відображенням результату, що покращує користувацький досвід.

Крім того, on-device підхід забезпечує незалежність від мережевого з’єднання, що розширює можливості використання мобільного застосунку. Користувач може здійснювати розпізнавання мовлення навіть за відсутності доступу до Інтернету або в умовах нестабільного з’єднання. Це особливо актуально для мобільних сценаріїв використання, де стабільність мережі не гарантується.

1.2.2 Вплив обмежених ресурсів мобільних платформ на моделі

Мобільні пристрої мають обмежені обчислювальні ресурси у порівнянні з серверними системами, що безпосередньо впливає на вибір і реалізацію моделей розпізнавання мовлення. Обмеження стосуються продуктивності центрального процесора, обсягу оперативної пам’яті та енергоспоживання. У зв’язку з цим використання великих і складних

моделей без оптимізації є недоцільним, оскільки це може призвести до значних затримок та швидкого розряду батареї.

Для забезпечення ефективної роботи систем розпізнавання мовлення на мобільних пристроях застосовуються методи оптимізації моделей. До них належать квантизація ваг, зменшення розрядності обчислень, обрізання незначущих параметрів та використання компактних архітектур. Такі підходи дозволяють зменшити розмір моделі та прискорити її виконання без суттєвої втрати точності розпізнавання.

Важливим аспектом є також використання апаратного прискорення, яке доступне у сучасних мобільних пристроях. Зокрема, у середовищі iOS моделі можуть виконуватись із залученням графічного процесора або нейронного співпроцесора через фреймворк Core ML. Це дозволяє досягти балансу між продуктивністю та енергоефективністю, що є критичним для реалізації розпізнавання мовлення у реальному часі на пристрої користувача.

1.2.3 Порівняння on-device та cloud-based підходів

Існують два основних підходи до реалізації систем розпізнавання мовлення – локальний (on-device) та хмарний (cloud-based), які відрізняються способом обробки даних і архітектурою системи. У випадку on-device підходу всі обчислення виконуються безпосередньо на мобільному пристрої користувача, тоді як cloud-based рішення передбачають передачу аудіосигналу на віддалені сервери для подальшої обробки. Порівняння цих підходів дозволяє визначити їхні переваги та обмеження у контексті розробки мобільних застосунків (рисунок 1.6).

Локальний підхід забезпечує високу швидкість обробки та мінімальні затримки, оскільки відсутня необхідність передачі даних мережею. Крім того, він гарантує підвищений рівень конфіденційності, оскільки аудіодані не залишають пристрій користувача. Важливою перевагою є також

можливість роботи у режимі offline, що дозволяє використовувати застосунок незалежно від наявності доступу до Інтернету. Проте такі системи обмежені ресурсами пристрою, що впливає на складність моделей та їхню точність.



Рисунок 1.6 – Порівняння on-device та cloud-based підходів

У свою чергу, cloud-based підхід дозволяє використовувати більш потужні обчислювальні ресурси та великі моделі, що позитивно впливає на точність розпізнавання (рисунок 1.6). Хмарні системи легко масштабуються та можуть швидко оновлювати моделі без необхідності оновлення самого застосунку. Однак цей підхід має суттєві недоліки, серед яких затримки, пов’язані з передачею даних, залежність від стабільності мережевого з’єднання та потенційні ризики для конфіденційності користувача. Таким чином, вибір між on-device та cloud-based підходами залежить від вимог до системи, зокрема щодо швидкодії, точності, конфіденційності та доступності.

1.3 Моделі та алгоритми для розпізнавання мовлення

Моделі та алгоритми розпізнавання мовлення є основою функціонування сучасних систем автоматичної транскрипції аудіосигналу. Вони забезпечують перетворення акустичних характеристик мовлення у текстове представлення шляхом послідовного застосування методів обробки сигналів і машинного навчання. Загальний процес роботи таких моделей включає етапи обробки сигналу, виділення ознак, застосування акустичної моделі, мовного моделювання та декодування результату, що дозволяє отримати фінальний текст із урахуванням контексту та мовних закономірностей (рисунок 1.7).

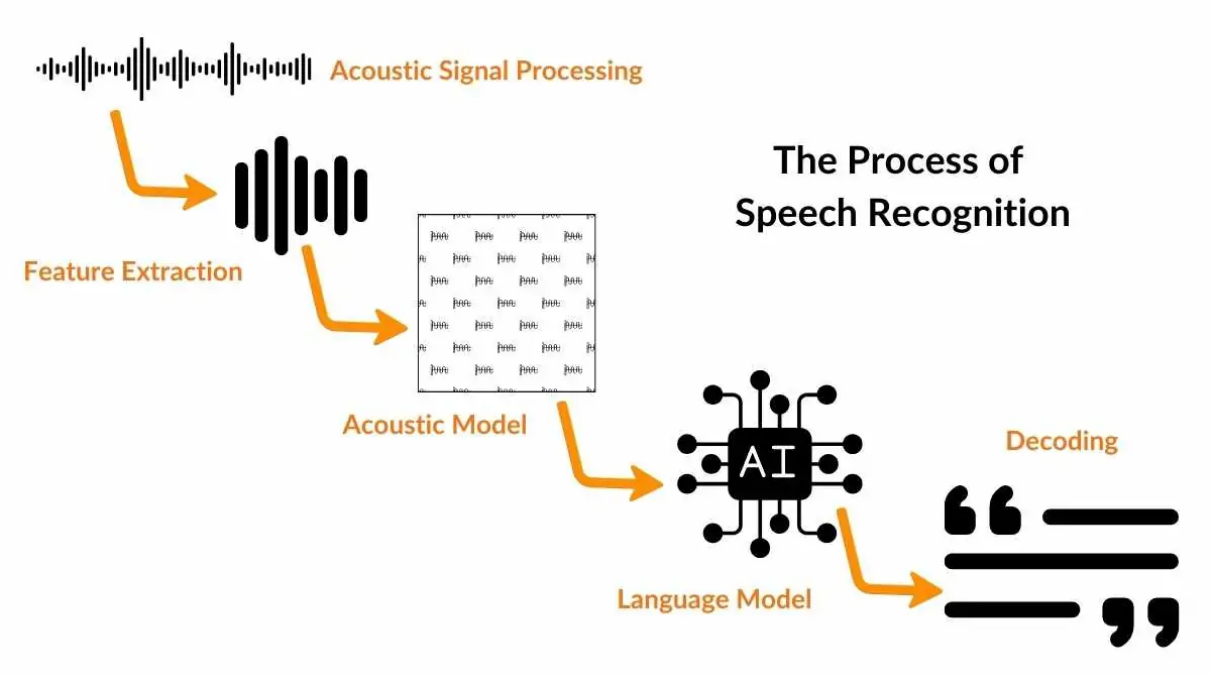


Рисунок 1.7 – Процес роботи моделей розпізнавання мовлення

На початкових етапах, відповідно до наведеної схеми (рисунок 1.7), аудіосигнал проходить обробку та перетворюється у набір ознак, які відображають його частотні та часові характеристики. Далі ці ознаки передаються до акустичної моделі, яка встановлює відповідність між

сигналом і базовими мовними одиницями, такими як фонеми або субсловні елементи. Акустичні моделі можуть реалізовуватись як у вигляді класичних статистичних підходів, так і з використанням глибоких нейронних мереж.

Після цього застосовується мовна модель, яка враховує контекст та ймовірнісні залежності між словами (рисунок 1.7). Це дозволяє скоригувати результат розпізнавання та підвищити його точність, особливо у випадках неоднозначного звучання. Завершальним етапом є декодування, під час якого формується остаточна текстова послідовність. Таким чином, поєднання різних моделей та алгоритмів дозволяє створювати ефективні системи розпізнавання мовлення, здатні працювати у реальному часі навіть в умовах обмежених ресурсів мобільних пристроїв.

1.3.1 Класичні підходи: HMM та GMM

Класичні підходи до автоматичного розпізнавання мовлення базуються на статистичних методах, серед яких найбільш поширеними є моделі прихованих марковських процесів та гаусові суміші. Модель прихованих марковських процесів використовується для опису послідовності мовних одиниць як стохастичного процесу з прихованими станами, що відповідають фонемам або їх частинам. Основною перевагою цього підходу є здатність моделювати часову структуру мовлення та враховувати послідовність сигналу.

Для опису розподілу акустичних ознак у межах кожного стану широко застосовуються гаусові суміші. Вони дозволяють апроксимувати складні розподіли ознак за допомогою комбінації кількох нормальних розподілів, що забезпечує гнучкість моделі. Поєднання HMM та GMM стало стандартом у класичних системах розпізнавання мовлення, оскільки дозволяло ефективно вирішувати задачу зіставлення акустичних сигналів із мовними одиницями.

Незважаючи на свою ефективність, такі підходи мають ряд обмежень, пов’язаних із необхідністю ручного виділення ознак та обмеженою здатністю моделювати складні нелінійні залежності у мовленні. Крім того, якість розпізнавання значною мірою залежить від припущень про незалежність ознак та простоту їх розподілів. Це стало передумовою для переходу до більш потужних методів, заснованих на глибокому навчанні.

1.3.2 Глибокі нейронні мережі у задачах ASR

Глибокі нейронні мережі стали основним інструментом у сучасних системах автоматичного розпізнавання мовлення завдяки їх здатності автоматично виявляти складні залежності у даних. На відміну від класичних підходів, вони дозволяють виконувати як виділення ознак, так і моделювання мовлення в межах єдиної архітектури. Це значно спрощує процес побудови системи та підвищує точність розпізнавання.

У задачах ASR використовуються різні типи нейронних мереж, зокрема рекурентні нейронні мережі, згорткові мережі та трансформери. Рекурентні мережі ефективно враховують часову залежність сигналу, тоді як трансформерні архітектури забезпечують більш гнучке моделювання контексту за рахунок механізму уваги. Такі моделі дозволяють працювати з довгими послідовностями та забезпечують високу точність навіть у складних акустичних умовах.

Сучасні підходи також включають використання наскрізних моделей, які безпосередньо перетворюють аудіосигнал у текст без явного поділу на окремі етапи. Це дозволяє оптимізувати весь процес розпізнавання як єдину задачу та досягти кращих результатів. У контексті мобільних застосунків важливим є також питання оптимізації таких моделей для роботи на пристрої, що досягається шляхом зменшення їх розміру та використання спеціалізованих інструментів, таких як Core ML.

1.3.3 Архітектура моделей типу encoder-decoder та трансформери

Моделі типу encoder-decoder є одним із ключових підходів у задачах автоматичного розпізнавання мовлення, оскільки дозволяють безпосередньо перетворювати вхідний аудіосигнал у текстову послідовність. Основна ідея цього підходу полягає у розділенні процесу на два етапи: кодування вхідних даних у внутрішнє представлення та декодування цього представлення у вихідну послідовність. Така архітектура забезпечує гнучкість у роботі з послідовностями різної довжини та дозволяє ефективно враховувати контекст під час генерації тексту.

У сучасних системах розпізнавання мовлення широко застосовуються трансформерні моделі, які реалізують підхід encoder-decoder із використанням механізму уваги. Структура такої моделі включає енкодер, що обробляє вхідні ознаки мовлення, та декодер, який послідовно генерує текстові токени (рисунок 1.8). Ключовим елементом є механізм багатоголової уваги, що дозволяє моделі визначати найбільш релевантні частини вхідної послідовності для кожного кроку декодування. Додатково використовуються позиційні кодування, які забезпечують урахування порядку елементів у послідовності.

Перевагою трансформерних архітектур є їх здатність ефективно обробляти довгі послідовності та паралелізувати обчислення, що значно підвищує продуктивність у порівнянні з рекурентними моделями. Крім того, такі моделі демонструють високу точність завдяки можливості врахування глобального контексту.

У задачах розпізнавання мовлення це дозволяє краще відтворювати складні мовні конструкції та зменшувати кількість помилок. У контексті on-device систем важливим аспектом є оптимізація трансформерних моделей для роботи на мобільних пристроях.

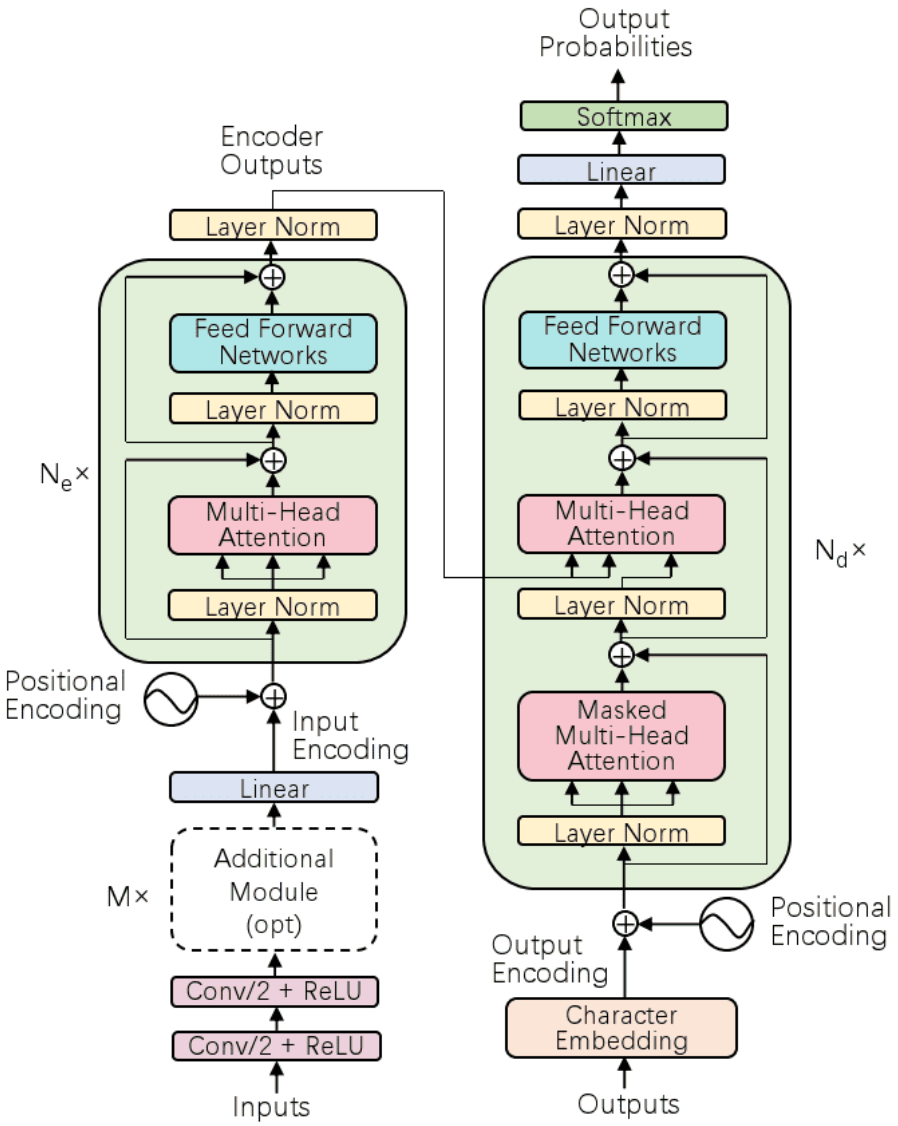


Рисунок 1.8 – Архітектура трансформерної моделі типу encoder-decoder

Це включає зменшення кількості параметрів, використання ефективних реалізацій механізму уваги та застосування апаратного прискорення.

Незважаючи на високу обчислювальну складність, сучасні оптимізовані трансформерні моделі, зокрема ті, що використовуються у Whisper, дозволяють реалізувати розпізнавання мовлення у реальному часі безпосередньо на пристрої користувача.

1.3.4 Особливості моделі Whisper для розпізнавання мовлення

Модель Whisper є сучасною трансформерною моделлю для автоматичного розпізнавання мовлення, яка реалізує підхід encoder-decoder та призначена для роботи з різними мовами та акустичними умовами. Її особливістю є використання єдиної універсальної архітектури, яка здатна виконувати не лише транскрипцію мовлення. Вхідними даними для моделі є логарифмічна мел-спектрограма аудіосигналу, що дозволяє ефективно представляти частотні характеристики мовлення. Архітектурно модель складається з енкодера, який обробляє спектрограму, та декодера, що генерує текстову послідовність (рисунок 1.9).

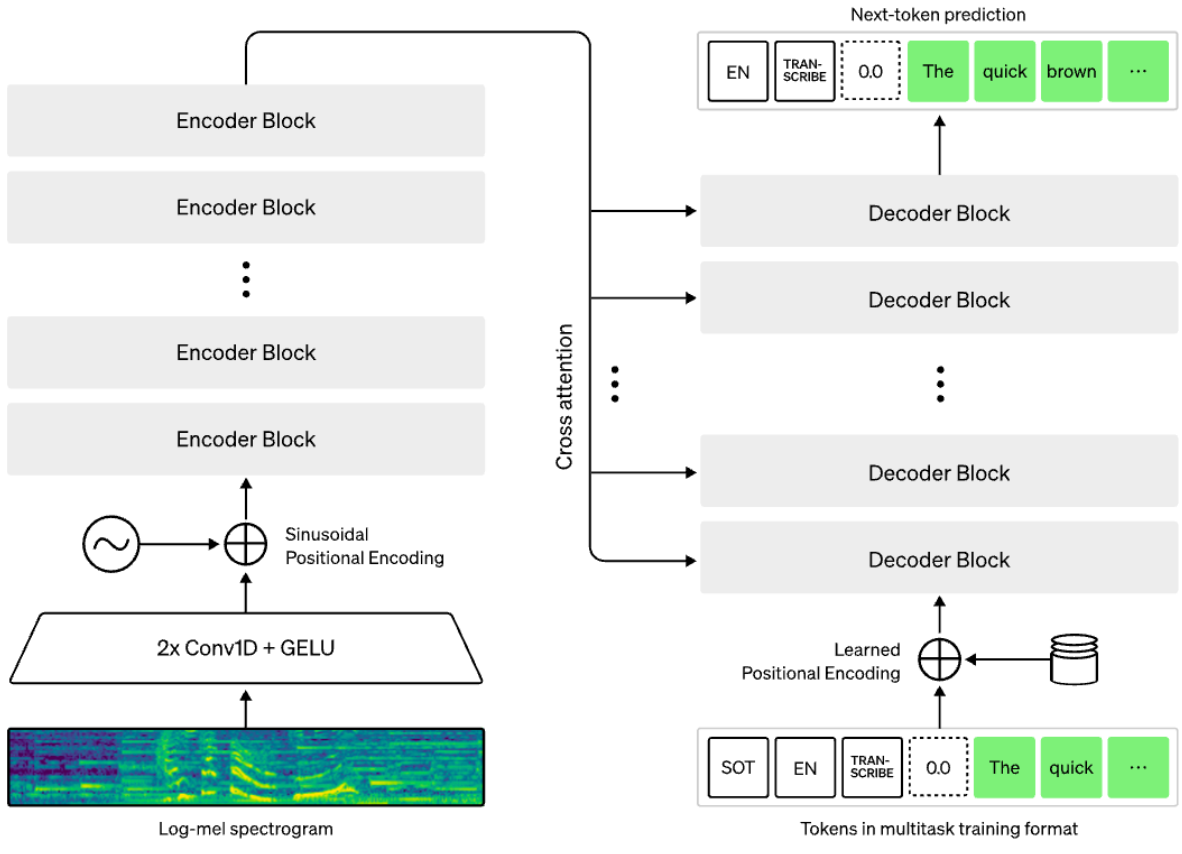


Рисунок 1.9 – Архітектура моделі Whisper для розпізнавання мовлення

Енкодер включає кілька блоків трансформера, які формують узагальнене представлення аудіосигналу, тоді як декодер використовує механізм крос-уваги для поєднання цього представлення з уже згенерованими токенами. Особливістю Whisper є використання спеціальних токенів керування, які визначають режим роботи моделі, наприклад, мову або задачу транскрипції.

Важливою перевагою моделі Whisper є її здатність працювати з шумними та реальними аудіоданими завдяки навчанню на великому різноманітному датасеті. Це дозволяє забезпечити високу якість розпізнавання у різних умовах, включаючи фоновий шум, різні акценти та швидкість мовлення. Крім того, модель демонструє хорошу узагальнювальну здатність без необхідності додаткового налаштування під конкретні умови використання.

У контексті мобільних застосунків важливою є можливість оптимізації Whisper для виконання на пристрої (рисунок 1.9). Для цього використовуються спрощені або квантизовані версії моделі, а також спеціалізовані бібліотеки, такі як ggml, що дозволяють ефективно виконувати обчислення на мобільних процесорах. Поєднання Whisper із фреймворком Core ML відкриває можливість реалізації розпізнавання мовлення у реальному часі без використання хмарних сервісів, що відповідає вимогам сучасних мобільних застосунків щодо швидкодії та конфіденційності.

1.4 Технологічне забезпечення iOS для роботи з мовленням

Технологічне забезпечення платформи iOS надає широкий набір інструментів для реалізації функціональності, пов’язаної з обробкою мовлення. Серед них важливе місце займають фреймворки для роботи з аудіо, машинним навчанням та обробкою природної мови. Це дозволяє

створювати ефективні мобільні застосунки, здатні виконувати розпізнавання мовлення у реальному часі без залучення зовнішніх сервісів.

Одним із ключових компонентів є AVFoundation, який забезпечує захоплення аудіосигналу з мікрофона, його обробку та керування аудіосесіями. Саме цей фреймворк використовується для реалізації потокового отримання аудіоданих, що є необхідним для систем розпізнавання мовлення у реальному часі. Додатково платформа iOS пропонує фреймворк Speech (рисунок 1.10), який дозволяє виконувати розпізнавання мовлення, проте у більшості випадків він орієнтований на використання хмарних сервісів, що обмежує його застосування у задачах із підвищеними вимогами до конфіденційності.

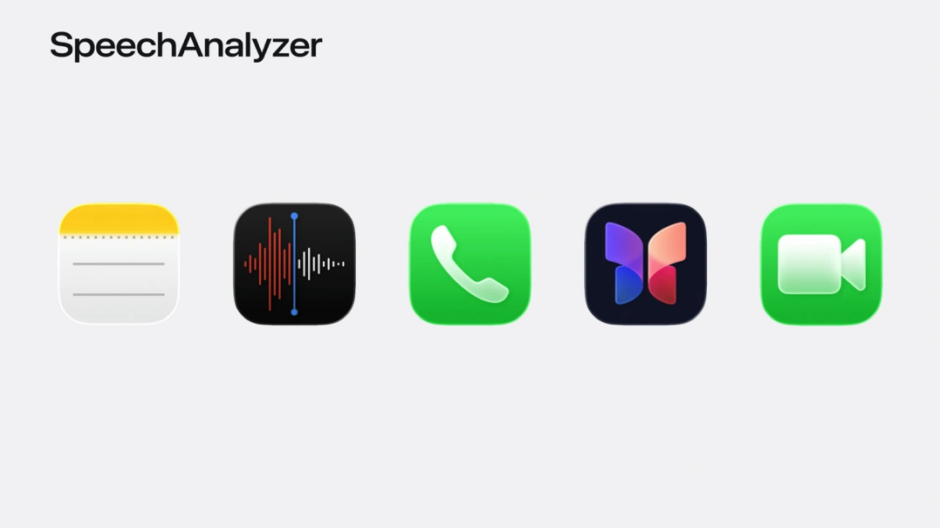


Рисунок 1.10 – Фреймворк SpeechAnalyzer

Для реалізації on-device розпізнавання мовлення ключову роль відіграє фреймворк Core ML, який забезпечує інтеграцію та виконання моделей машинного навчання безпосередньо на пристрої. Завдяки підтримці апаратного прискорення, зокрема нейронного процесора, Core ML дозволяє ефективно виконувати складні моделі, такі як Whisper, із мінімальними затримками. Це робить можливим створення застосунків, що

працюють у режимі реального часу та не залежать від мережевого з’єднання. Крім того, важливим елементом є використання сучасних технологій побудови інтерфейсу, зокрема SwiftUI, що дозволяє реалізувати зручний та інтерактивний інтерфейс для роботи з мовленням. Таким чином, платформа iOS надає всі необхідні засоби для реалізації ефективних систем розпізнавання мовлення на пристрої користувача.

1.4.1 Можливості фреймворків AVFoundation та Speech

Фреймворк AVFoundation є базовим інструментом платформи iOS для роботи з аудіо та відео, що забезпечує доступ до апаратних можливостей пристрою. У контексті розпізнавання мовлення він використовується для захоплення аудіосигналу з мікрофона, налаштування аудіосесії та керування потоками аудіоданих. Завдяки використанню AVAudioEngine можливо реалізувати безперервне потокове отримання аудіо.

Однією з важливих переваг AVFoundation є можливість гнучкого налаштування параметрів аудіозапису, таких як частота дискретизації, формат сигналу та буферизація. Це дозволяє адаптувати систему під вимоги моделі розпізнавання мовлення, зокрема забезпечити відповідність формату аудіоданих. Крім того, фреймворк підтримує обробку аудіо в реальному часі, що дає змогу передавати дані безпосередньо до моделі машинного навчання без значних затримок.

Фреймворк Speech, у свою чергу, надає готові інструменти для розпізнавання мовлення, включаючи перетворення аудіо у текст. Він підтримує як потокове, так і пакетне розпізнавання, а також інтеграцію з системними сервісами iOS. Однак у більшості випадків цей фреймворк використовує хмарні ресурси для обробки мовлення, що обмежує його застосування у задачах, де необхідна повна автономність та конфіденційність даних.

1.4.2 Використання Core ML для інтеграції моделей машинного навчання

Core ML є основним фреймворком платформи iOS для інтеграції та виконання моделей машинного навчання безпосередньо на пристрої. Він дозволяє використовувати попередньо натреновані моделі, конвертовані у спеціальний формат, оптимізований для мобільних пристроїв. У задачах розпізнавання мовлення Core ML виступає ключовим компонентом, що забезпечує виконання моделей у режимі реального часу.

Однією з важливих переваг Core ML є підтримка апаратного прискорення, що дозволяє ефективно використовувати ресурси центрального процесора, графічного процесора та нейронного співпроцесора. Це значно підвищує швидкість обчислень та знижує енергоспоживання, що є критично важливим для мобільних застосунків. Завдяки цьому можливо реалізувати складні моделі розпізнавання мовлення, зокрема трансформерні архітектури, без суттєвого впливу на продуктивність пристрою.

Core ML також забезпечує зручні механізми інтеграції моделей у застосунок, включаючи автоматичну генерацію інтерфейсів для роботи з моделлю та підтримку обробки потокових даних. Це дозволяє поєднати моделі машинного навчання з іншими компонентами системи, такими як AVFoundation, для реалізації повного конвеєра обробки мовлення. У результаті досягається ефективна взаємодія між захопленням аудіо, його обробкою та генерацією тексту.

1.4.3 Оптимізація моделей для роботи на мобільних пристроях

Оптимізація моделей машинного навчання є необхідною умовою їх ефективного використання на мобільних пристроях, які мають обмежені обчислювальні ресурси. Великі моделі, що забезпечують високу точність

розпізнавання мовлення, часто мають значний розмір і складність, що ускладнює їх використання без оптимізації. Тому важливим завданням є зменшення обсягу моделі та прискорення її виконання.

Одним із основних методів оптимізації є квантизація, яка передбачає зменшення розрядності числових параметрів моделі. Це дозволяє значно зменшити обсяг пам’яті та підвищити швидкість обчислень без суттєвої втрати точності. Додатково застосовуються методи обрізання параметрів та спрощення архітектури моделі, що дозволяє зменшити кількість обчислень та зробити модель більш придатною для мобільного середовища.

Також важливим аспектом є використання апаратного прискорення та спеціалізованих бібліотек, які дозволяють ефективно виконувати моделі на мобільних пристроях. У середовищі iOS це досягається за рахунок інтеграції з Core ML та використання нейронного процесора. Завдяки комплексному підходу до оптимізації стає можливим реалізувати розпізнавання мовлення у реальному часі без перевантаження пристрою, що є ключовою вимогою для сучасних мобільних застосунків.

1.5 Аналіз сценаріїв використання мобільного застосунку

Аналіз сценаріїв використання мобільного застосунку є важливим етапом проєктування системи, оскільки дозволяє визначити ключові способи взаємодії користувача із функціональністю застосунку. У контексті системи розпізнавання мовлення основна увага приділяється сценаріям, пов’язаним із записом аудіо, його обробкою у реальному часі та подальшим збереженням результатів. Це дозволяє сформувати вимоги до інтерфейсу, продуктивності та поведінки системи.

Застосунок, що розробляється, орієнтований на створення текстових нотаток на основі мовлення користувача, що визначає основні сценарії його використання. Користувач взаємодіє із системою через простий інтерфейс, який дозволяє розпочати та завершити запис, спостерігати процес

транскрипції та зберігати отриманий результат. Особливістю є необхідність забезпечення безперервної роботи системи у режимі реального часу без значних затримок.

Аналіз сценаріїв також дозволяє врахувати різні умови використання застосунку, зокрема наявність шумів, нестабільність мережевого з’єднання або обмеження ресурсів пристрою. Це впливає на вибір технологій та архітектурних рішень, зокрема на доцільність використання on-device підходу. Таким чином, детальне дослідження сценаріїв використання є основою для формування вимог до системи та забезпечення її ефективної роботи.

1.5.1 Основні сценарії взаємодії користувача із системою

Основним сценарієм використання застосунку є запис мовлення користувача у реальному часі з подальшим автоматичним перетворенням у текст. Користувач ініціює процес запису через інтерфейс застосунку, після чого система починає безперервно отримувати аудіосигнал та обробляти його. Результат розпізнавання відображається динамічно у вигляді тексту, що поступово доповнюється у процесі мовлення.

Іншим важливим сценарієм є збереження отриманих транскрипцій у вигляді текстових нотаток. Після завершення запису користувач має можливість зберегти результат, переглянути його або використати у подальшій роботі. Це передбачає реалізацію механізмів локального збереження даних, а також забезпечення зручного доступу до раніше створених нотаток.

Додаткові сценарії включають редагування тексту, пошук серед збережених нотаток та роботу у режимі offline. Користувач може взаємодіяти із застосунком у будь-яких умовах, незалежно від наявності мережевого з’єднання.

1.5.2 Вимоги до обробки мовлення в реальному часі

Обробка мовлення в реальному часі висуває жорсткі вимоги до продуктивності системи, оскільки затримки у відображенні результату можуть негативно впливати на користувацький досвід. Система повинна забезпечувати мінімальну затримку між моментом вимови та відображенням тексту, що досягається за рахунок ефективної обробки аудіоданих та оптимізації моделей. Важливим є також забезпечення стабільної роботи при безперервному потоці даних.

Ще однією важливою вимогою є точність розпізнавання мовлення, особливо в умовах шуму або варіативності вимови. Система повинна коректно інтерпретувати мовлення користувача та мінімізувати кількість помилок у тексті. Для цього використовуються сучасні моделі машинного навчання, здатні враховувати контекст та адаптуватися до різних умов використання.

Крім того, обробка мовлення в реальному часі повинна бути енергоефективною та не створювати надмірного навантаження на пристрій. Це особливо важливо для мобільних застосунків, які працюють тривалий час. Досягнення цього балансу потребує використання оптимізованих моделей, апаратного прискорення та ефективного управління ресурсами системи.

1.6 Формування вимог до програмної системи

Формування вимог до програмної системи є ключовим етапом розробки мобільного застосунку, оскільки визначає його функціональні можливості, обмеження та критерії якості. У контексті системи розпізнавання мовлення вимоги формуються з урахуванням специфіки обробки аудіосигналів, роботи у реальному часі та необхідності забезпечення високої точності результатів.

Особливу увагу приділено узгодженню функціональних і нефункціональних вимог, оскільки ефективність застосунку залежить не лише від наявності необхідних функцій, але й від швидкодії, стабільності та безпеки. У межах даної роботи вимоги формуються з урахуванням on-device підходу, що передбачає виконання обчислень безпосередньо на пристрої користувача. Це накладає додаткові обмеження на ресурси системи та визначає необхідність оптимізації всіх компонентів.

1.6.1 Функціональні вимоги до застосунку

Функціональні вимоги визначають основні можливості застосунку, пов’язані з обробкою мовлення та взаємодією з користувачем. До них належить реалізація запису мовлення у реальному часі, автоматичне перетворення аудіосигналу у текст, а також динамічне відображення результату під час мовлення. Важливою функцією є можливість збереження отриманих транскрипцій у вигляді текстових нотаток для подальшого використання.

Крім основних функцій, застосунок повинен забезпечувати перегляд раніше створених нотаток, а також можливість їх редагування та пошуку. Додатково передбачається підтримка роботи у режимі offline, що дозволяє використовувати систему без доступу до мережі Інтернет. Сукупність цих функцій формує повноцінний інструмент для роботи з мовленням у мобільному середовищі.

1.6.2 Нефункціональні вимоги: продуктивність, затримка, енергоефективність

Нефункціональні вимоги визначають якісні характеристики системи, які безпосередньо впливають на її ефективність та зручність використання. Однією з ключових вимог є висока продуктивність, що забезпечує обробку

аудіосигналу у режимі реального часу без значних затримок. Затримка між мовленням користувача та відображенням тексту повинна бути мінімальною, що досягається шляхом оптимізації алгоритмів та використання апаратного прискорення.

Важливим аспектом є також енергоефективність системи, оскільки мобільні пристрої мають обмежені ресурси акумулятора. Застосунок повинен мінімізувати споживання енергії під час безперервної обробки мовлення, що потребує використання оптимізованих моделей та ефективного управління ресурсами. Забезпечення балансу між продуктивністю та енергоефективністю є одним із ключових завдань при розробці системи.

1.6.3 Вимоги до конфіденційності та безпеки даних

Конфіденційність та безпека даних є критично важливими аспектами для застосунків, що працюють із мовленням користувача. Оскільки аудіодані можуть містити особисту або чутливу інформацію, необхідно забезпечити їх захист від несанкціонованого доступу.

Використання on-device підходу дозволяє уникнути передачі даних на зовнішні сервери, що значно знижує ризики витоку інформації. Додатково система повинна забезпечувати безпечне збереження текстових нотаток на пристрої користувача, використовуючи вбудовані механізми захисту платформи iOS.

1.7 Постановка задачі

У межах даної роботи ставиться задача розробки мобільного застосунку для операційної системи iOS, призначеного для розпізнавання мовлення та перетворення його у текст у режимі реального часу. Система повинна забезпечувати можливість запису аудіосигналу з мікрофона, його

подальшу обробку та відображення результатів транскрипції без затримок, а також збереження отриманих текстових даних для подальшого використання.

Реалізація інтелектуальних функцій передбачається безпосередньо на мобільному пристрої з використанням технології Core ML та інтегрованих моделей розпізнавання мовлення, що дозволяє виконувати обчислення локально без передачі даних на зовнішні сервери. Це забезпечує підвищення рівня конфіденційності, зменшення залежності від мережевого з’єднання та покращення швидкодії системи.

Реалізація такого підходу дозволить створити ефективний інструмент для роботи з мовленням у мобільному середовищі, який поєднує високу точність розпізнавання, мінімальну затримку та енергоефективність. Запропоноване рішення розширює можливості традиційних застосунків для запису звуку, доповнюючи їх функціональністю інтелектуального аналізу мовлення та автоматичної транскрипції.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Теоретичні основи цифрового подання мовленнєвого сигналу

Мовлення людини є складним акустичним сигналом, який формується внаслідок роботи голосового апарату та поширюється у вигляді механічних коливань повітряного середовища. Для подальшої комп’ютерної обробки аналоговий мовленнєвий сигнал повинен бути перетворений у цифрову форму, придатну для аналізу, збереження та використання алгоритмами машинного навчання. Саме цифрове подання мовлення є базовою основою функціонування сучасних систем автоматичного розпізнавання мовлення, оскільки нейронні мережі не працюють безпосередньо з аналоговими хвилями, а використовують їх математичне представлення у вигляді послідовностей числових значень. У сучасних системах ASR якість цифрового подання сигналу безпосередньо впливає на точність подальшого розпізнавання, стабільність роботи моделей та швидкість обробки аудіопотоку.

У задачах on-device розпізнавання мовлення питання цифрової обробки сигналу набуває ще більшої важливості через обмежені обчислювальні ресурси мобільних пристроїв. Для ефективного функціонування системи необхідно забезпечити баланс між якістю аудіосигналу, швидкістю його обробки та обсягом обчислень, які виконуються під час реального часу. Саме тому сучасні системи розпізнавання мовлення використовують спеціальні методи дискретизації, спектрального аналізу та перетворення сигналу у компактні частотно-часові представлення. У межах даного підрозділу розглядаються теоретичні основи цифрового подання мовленнєвого сигналу, принципи його дискретизації та квантування, а також особливості представлення мовлення у часовій і частотній областях.

2.1.1 Фізична природа мовлення як акустичного сигналу

Мовлення людини з фізичної точки зору являє собою акустичний сигнал, який виникає внаслідок коливань голосових зв’язок та подальшого проходження звукових хвиль через мовленнєвий тракт. У процесі вимови утворюються зміни тиску повітря, які поширюються у просторі у вигляді хвиль та сприймаються слуховою системою людини (рисунок 2.1). Саме ці коливання формують аналоговий мовленнєвий сигнал, який надалі може бути зафіксований мікрофоном та перетворений у цифровий формат для подальшої комп’ютерної обробки.

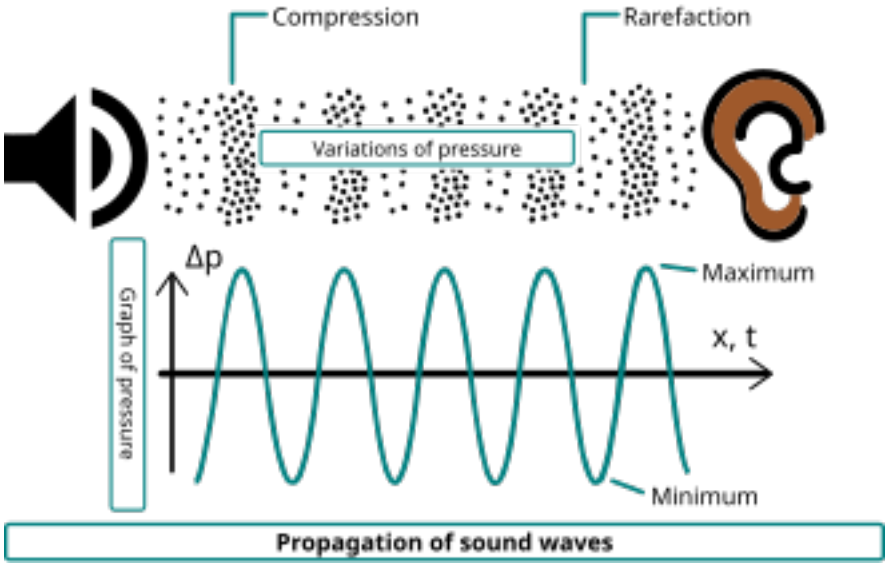


Рисунок 2.1 – Поширення мовленнєвого сигналу у вигляді акустичних хвиль

Акустичний мовленнєвий сигнал характеризується низкою фізичних параметрів, серед яких основними є амплітуда, частота та фаза коливань. Амплітуда визначає інтенсивність або гучність звуку, тоді як частота характеризує висоту тону мовлення. У реальних умовах мовленнєвий сигнал є складною суперпозицією великої кількості гармонічних компонент, що постійно змінюються в часі залежно від вимовлених звуків,

інтонації та темпу мовлення. Саме через складну структуру мовлення його обробка потребує використання спеціальних алгоритмів цифрової обробки сигналів та методів спектрального аналізу.

Особливістю мовленнєвого сигналу є його нестаціонарний характер, оскільки параметри сигналу постійно змінюються навіть протягом короткого проміжку часу. Це означає, що різні фонеми мають власні акустичні характеристики, а переходи між ними супроводжуються зміною спектрального складу сигналу. У сучасних системах автоматичного розпізнавання мовлення такі зміни аналізуються за допомогою часово-частотних представлень сигналу, які дозволяють виділяти характерні ознаки мовлення та використовувати їх як вхідні дані для нейронних мереж.

Під час роботи мобільних систем розпізнавання мовлення акустичний сигнал додатково піддається впливу зовнішніх шумів, реверберації приміщення, спотворень мікрофона та інших факторів, що можуть погіршувати якість розпізнавання. Саме тому на етапі попередньої обробки аудіо застосовуються методи нормалізації, фільтрації та приглушення шумів, які дозволяють підвищити якість вхідного сигналу перед його подальшим аналізом моделями машинного навчання.

2.1.2 Дискретизація, квантування та частота семплювання аудіосигналу

Для подальшої комп’ютерної обробки мовленнєвий сигнал повинен бути перетворений із неперервної аналогової форми у цифрову, а загальна послідовність такого аналого-цифрового перетворення (рисунок 2.2) включає етапи дискретизації, квантування та кодування сигналу. У системах автоматичного розпізнавання мовлення саме цифрове подання сигналу використовується як основа для подальшого спектрального аналізу.

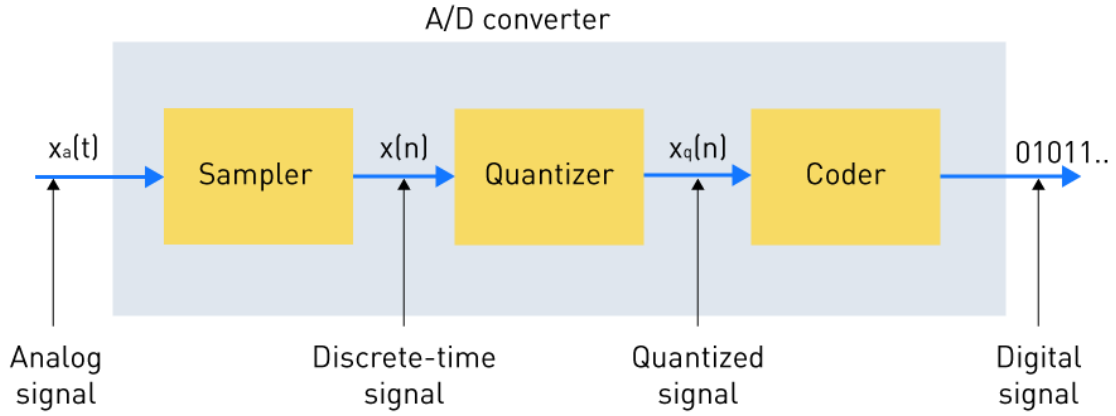


Рисунок 2.2 – Основні етапи аналого-цифрового перетворення аудіосигналу

Дискретизація полягає у вимірюванні значення аналогового сигналу через певні проміжки часу. Кількість таких вимірювань за одну секунду визначається частотою семплювання. Від значення частоти семплювання залежить точність відтворення мовленнєвого сигналу та можливість збереження його спектральних характеристик. Для задач розпізнавання мовлення найчастіше використовуються частоти 16 кГц або 44,1 кГц, оскільки вони дозволяють достатньо точно передавати мовленнєвий діапазон частот. Недостатня частота семплювання може призводити до втрати важливої акустичної інформації та погіршення точності розпізнавання.

Після дискретизації виконується квантування сигналу, під час якого неперервні значення амплітуди замінюються найближчими значеннями із наперед визначеного набору рівнів. У результаті цього процесу сигнал подається у вигляді числових значень, які можуть бути закодовані у двійковому форматі та оброблені цифровими системами. Кількість бітів, що використовується для квантування, визначає точність представлення сигналу та рівень шуму квантування. Збільшення бітової глибини дозволяє

зменшити втрати інформації, однак одночасно збільшує обсяг даних та навантаження на систему.

У задачах on-device розпізнавання мовлення параметри дискретизації та квантування мають особливе значення через необхідність забезпечення балансу між якістю аудіосигналу та швидкодією мобільного застосунку. Надто висока частота семплювання або велика бітова глибина можуть збільшувати обсяг обчислень і енергоспоживання пристрою, тоді як надто низькі значення здатні негативно впливати на точність роботи моделей розпізнавання мовлення. Саме тому сучасні мобільні системи ASR використовують оптимізовані параметри аудіообробки, які забезпечують прийнятну якість транскрипції при обмежених ресурсах пристрою.

2.1.3 Представлення мовлення у часовій та частотній областях

Мовленнєвий сигнал може аналізуватися як у часовій, так і у частотній області залежно від задачі обробки аудіоданих. У часовій області сигнал подається у вигляді зміни амплітуди в часі, що дозволяє оцінювати загальну структуру мовлення, тривалість звуків, паузи та інтенсивність сигналу. У частотній області аналізується спектральний склад сигналу, тобто набір частотних компонент, з яких формується мовлення. Відмінність між часовим та частотним представленням сигналу (рисунок 2.3) полягає у способі інтерпретації акустичних даних та виділення характеристик мовлення.

Часове представлення мовлення використовується переважно для первинного аналізу аудіосигналу та виявлення загальної структури мовленнєвого потоку. Саме у часовій області визначаються моменти початку та завершення мовлення, наявність пауз, зміни гучності та інші часові характеристики сигналу. Однак таке представлення не дозволяє повноцінно оцінити внутрішній спектральний склад мовлення.

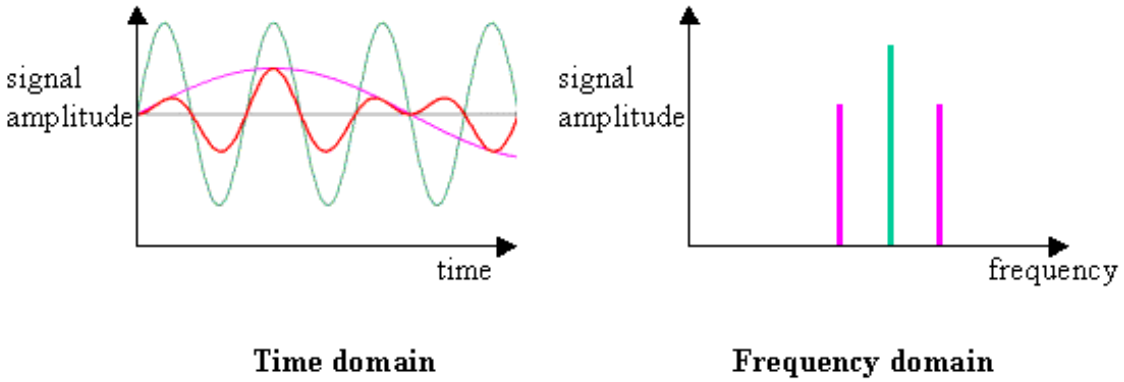


Рисунок 2.3 – Представлення сигналу у часовій та частотній областях

Для аналізу частотних характеристик мовлення застосовується спектральне подання сигналу, у якому визначаються частоти та їх енергетичний внесок у певний момент часу (рисунок 2.4).

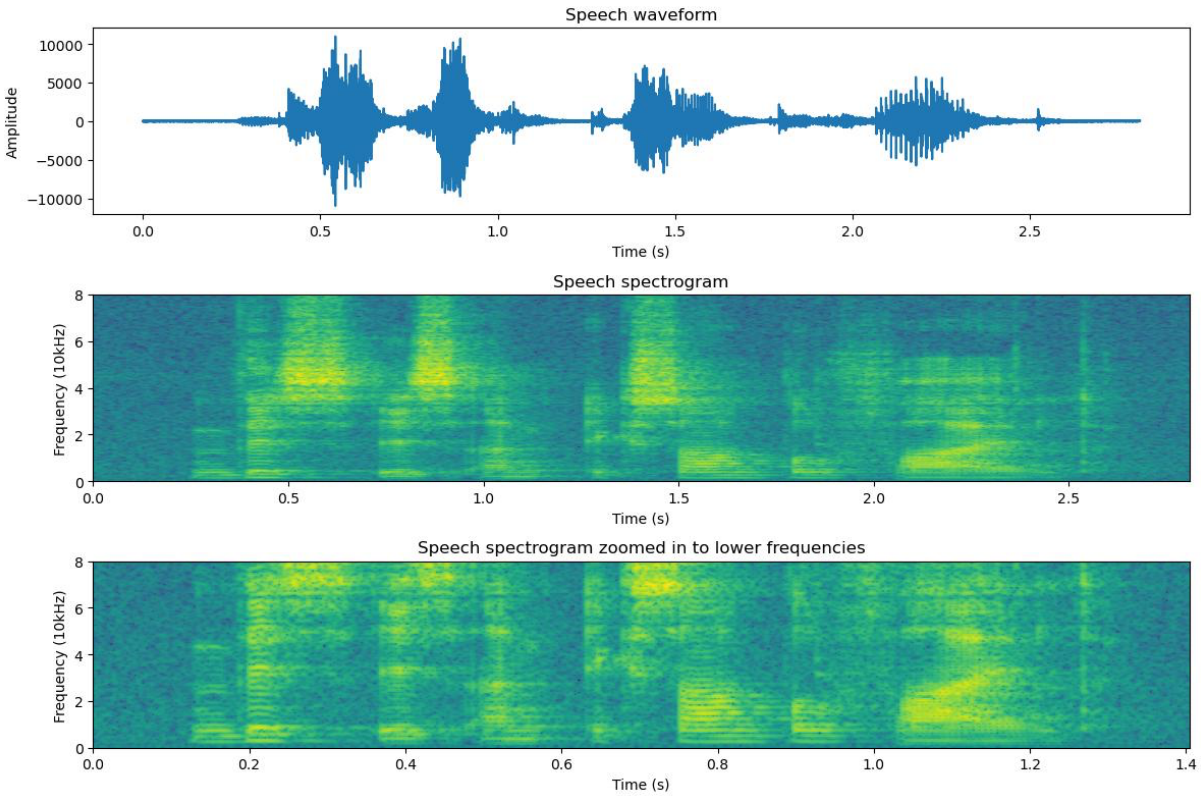


Рисунок 2.4 – Приклад часової форми мовлення та його спектрограми

Найбільш поширеним способом такого подання є спектрограма, яка відображає зміну частотного складу сигналу у часовому вимірі. Приклад часової форми мовлення та відповідної спектрограми (рисунок 2.4) демонструє взаємозв’язок між waveform-сигналом та його частотно-часовим поданням.

У сучасних системах автоматичного розпізнавання мовлення саме частотно-часове представлення сигналу використовується як основа для формування вхідних даних нейронних мереж. Зокрема, моделі типу Whisper працюють не безпосередньо з waveform-сигналом, а з його лог-мел-спектрограмою, яка містить компактне та інформативне подання мовлення.

Такий підхід дозволяє ефективніше виділяти мовленнєві ознаки, зменшувати вплив шумів та підвищувати точність розпізнавання мовлення у реальному часі.

2.2 Методи виділення ознак з аудіосигналу

Одним із ключових етапів автоматичного розпізнавання мовлення є виділення ознак з аудіосигналу. Безпосереднє використання сирого waveform-сигналу для роботи моделей машинного навчання є малоефективним через велику кількість надлишкової інформації та складність аналізу часових коливань. Саме тому у сучасних системах ASR застосовуються спеціальні методи перетворення мовлення у компактні та інформативні представлення, які дозволяють виділяти найбільш важливі акустичні характеристики сигналу. Отримані ознаки використовуються як вхідні дані для нейронних мереж та безпосередньо впливають на точність і стабільність процесу розпізнавання мовлення.

У сучасних системах розпізнавання мовлення найбільш поширеними є методи спектрального аналізу, які дозволяють досліджувати частотний склад мовлення у часовому вимірі. Особливу роль відіграють мел-спектрограми та лог-мел-спектрограми, оскільки саме вони

використовуються у трансформерних моделях типу Whisper. Крім того, ефективність виділення ознак значною мірою залежить від якості вхідного аудіосигналу, рівня шумів, тривалості аудіофрагментів та стабільності мовлення користувача. У межах даного підрозділу розглядаються основні методи спектрального аналізу мовленнєвого сигналу, принципи формування мел-спектрограм та фактори, що впливають на якість виділених ознак.

2.2.1 Спектральний аналіз мовленнєвого сигналу

Спектральний аналіз (рисунок 2.5) є одним із базових методів дослідження мовленнєвого сигналу, який використовується у сучасних системах автоматичного розпізнавання мовлення. Основною метою такого аналізу є визначення частотних компонент сигналу та оцінка їх зміни у часі. Мовлення людини складається з великої кількості звуків, кожен з яких має власний спектральний склад.

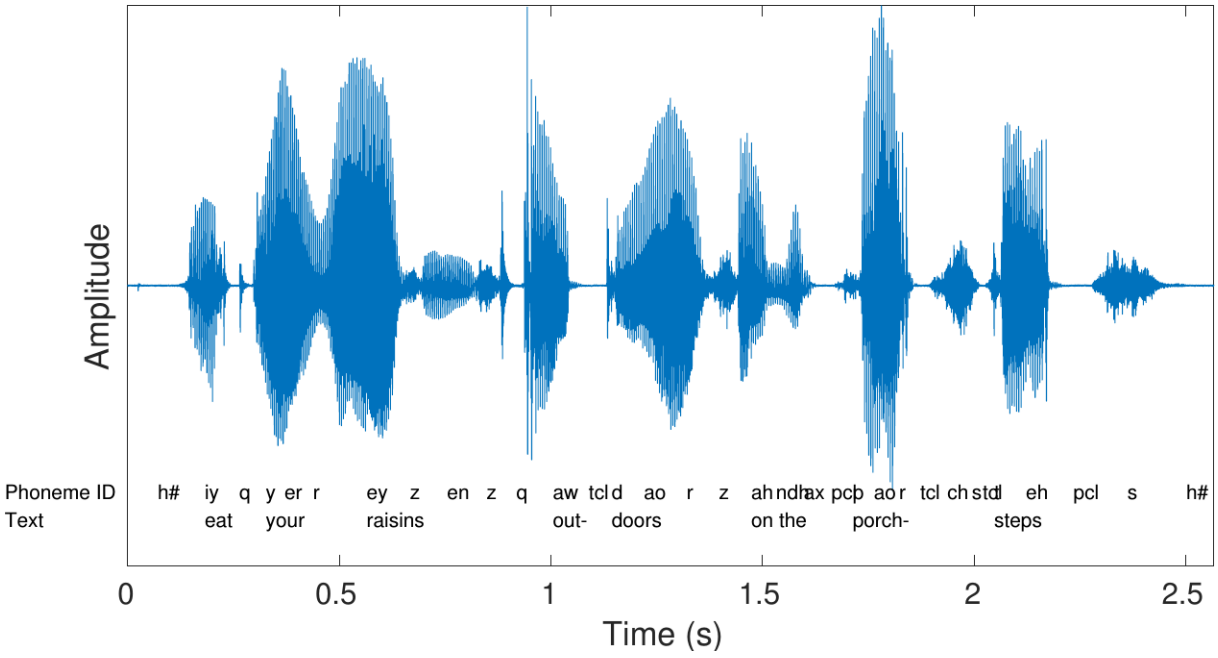


Рисунок 2.5 – Приклад часової структури мовленнєвого сигналу

Для виконання спектрального аналізу мовленнєвий сигнал зазвичай поділяється на невеликі часові фрагменти, після чого для кожного з них обчислюється частотний спектр. Такий підхід базується на припущенні, що протягом короткого інтервалу часу мовлення можна вважати умовно стаціонарним сигналом. Найбільш поширеним методом спектрального аналізу є швидке перетворення Фур'є, яке дозволяє визначити набір частот та їх енергетичні характеристики для окремого аудіофрагмента. У результаті формується спектральне представлення сигналу, придатне для подальшого аналізу алгоритмами машинного навчання.

Особливістю мовленнєвого сигналу є те, що різні фонеми мають власний спектральний профіль, який залежить від положення язика, губ, голосових зв'язок та інших елементів мовленнєвого апарату. Саме тому спектральний аналіз дозволяє ефективніше розділяти окремі звуки та формувати ознаки, які використовуються для автоматичного розпізнавання мовлення. У сучасних ASR-системах спектральні характеристики сигналу є значно інформативнішими за початкову waveform-форму, оскільки вони дозволяють компактно представити акустичну структуру мовлення.

У задачах on-device розпізнавання мовлення спектральний аналіз повинен виконуватись у режимі реального часу з мінімальними затримками та помірним використанням ресурсів пристрою. Саме тому сучасні мобільні системи використовують оптимізовані алгоритми обробки аудіо, які дозволяють швидко формувати спектральні ознаки без суттєвого навантаження на CPU, GPU або Neural Engine мобільного пристрою.

2.2.2 Мел-спектрограма як вхідне представлення для нейронних моделей

У сучасних системах автоматичного розпізнавання мовлення нейронні мережі зазвичай працюють не з початковим waveform-сигналом, а з його спектральним представленням у вигляді мел-спектрограми. Такий

підхід дозволяє значно зменшити обсяг вхідних даних та одночасно зберегти найбільш важливі акустичні характеристики мовлення. Мел-спектрограма формується шляхом перетворення звичайного спектра у спеціальну шкалу Mel, яка наближена до особливостей людського слухового сприйняття. Приклад мел-спектрограми мовленнєвого сигналу демонструє розподіл енергії частотних компонент у часі (рисунок 2.6).

Особливістю мел-шкали є нерівномірний розподіл частот, за якого нижні частоти аналізуються детальніше, ніж високі. Такий підхід обумовлений тим, що людський слух є більш чутливим саме до змін у низькочастотному діапазоні. У результаті мел-спектрограма дозволяє компактніше представляти мовленнєвий сигнал та ефективніше виділяти фонетичні особливості мовлення. Для додаткової стабілізації значень енергії у багатьох ASR-системах використовується логарифмічне масштабування, унаслідок чого формується лог-мел-спектрограма.

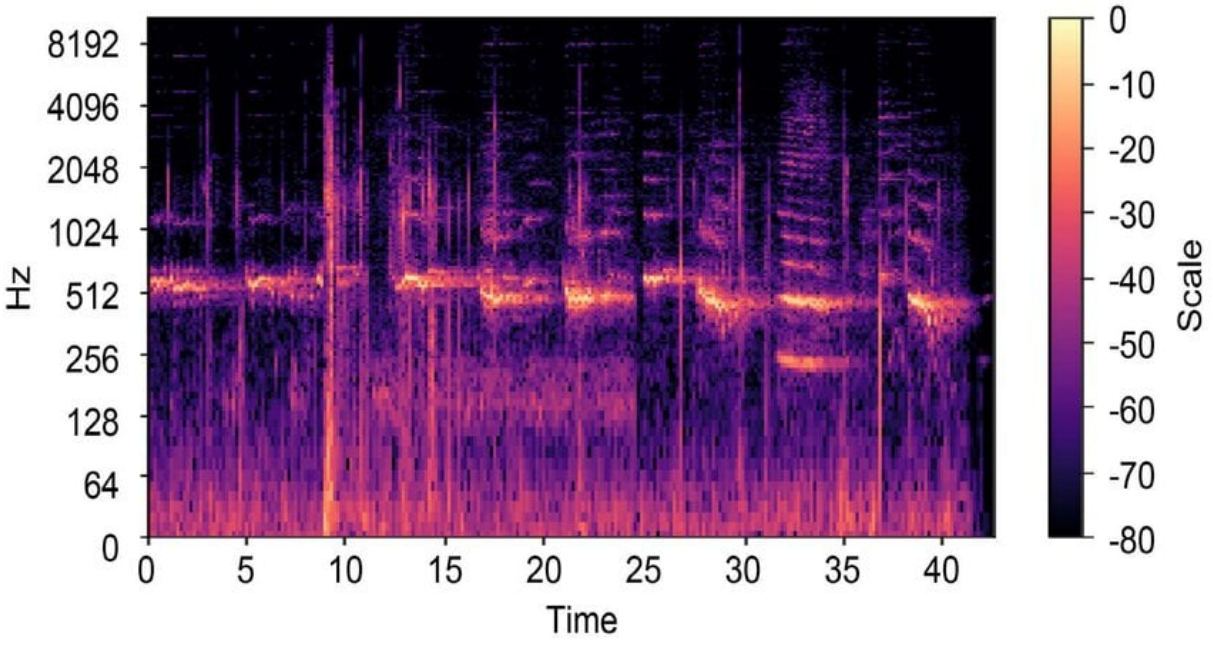


Рисунок 2.6 – Приклад мел-спектрограми мовленнєвого сигналу

Мел-спектрограми є основним типом вхідних даних для більшості сучасних моделей розпізнавання мовлення, зокрема трансформерних

архітектур типу Whisper. Загальна схема використання мел-спектрограми у процесі виділення ознак та подальшої класифікації (рисунок 2.7) демонструє, що після формування спектрального представлення сигнал подається на вхід нейронної мережі, яка виконує автоматичне виділення високорівневих ознак та прогнозування текстового результату.

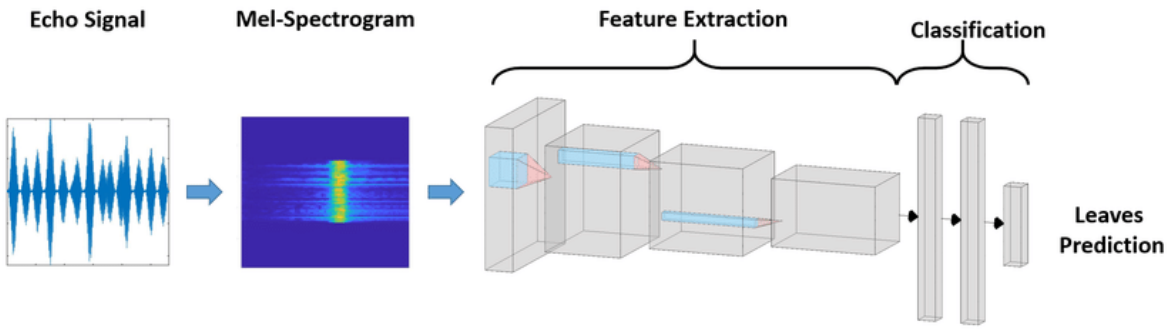


Рисунок 2.7 – Використання мел-спектрограми у процесі виділення ознак та класифікації

Використання мел-спектрограм дозволяє підвищити стійкість систем розпізнавання мовлення до шумів та варіацій голосу користувача. Крім того, спектральне представлення є значно компактнішим у порівнянні з waveform-сигналом, що особливо важливо для on-device систем, які працюють на мобільних пристроях з обмеженими ресурсами. Саме тому мел-спектрограми стали стандартним способом подання мовлення у сучасних ASR-моделях та системах обробки аудіосигналів.

2.2.3 Вплив шуму, пауз і довжини аудіофрагмента на якість розпізнавання

Якість автоматичного розпізнавання мовлення значною мірою залежить від характеристик вхідного аудіосигналу. Одними з основних

факторів, що впливають на точність роботи ASR-систем, є рівень шуму, наявність тривалих пауз та довжина аудіофрагментів. У реальних умовах використання мобільних застосунків мовлення часто супроводжується сторонніми звуками навколишнього середовища, змінами гучності та нестабільністю запису, що ускладнює процес виділення мовленнєвих ознак. Вплив попередньої обробки сигналу на структуру аудіоданих (рисунок 2.8) демонструє зміну waveform-сигналу після приглушення шумів та нормалізації аудіо.

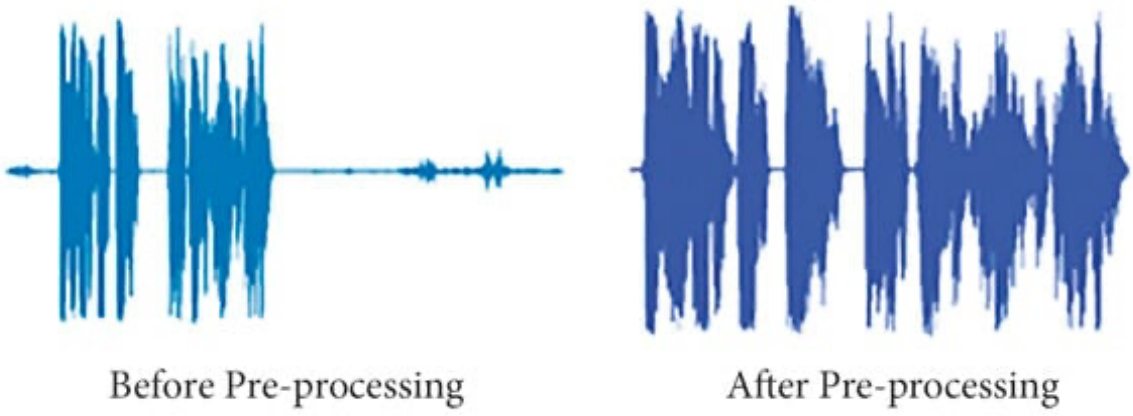


Рисунок 2.8 – Вплив попередньої обробки сигналу на структуру аудіоданих

Наявність шумів у сигналі може призводити до спотворення спектральних характеристик мовлення та помилкового виділення ознак нейронною мережею. Особливо критичними є фонові голоси, шум транспорту, вітер та реверберація приміщення, оскільки вони здатні перекривати корисний мовленнєвий сигнал. Для зменшення негативного впливу шумів у сучасних системах розпізнавання мовлення використовуються методи фільтрації, нормалізації амплітуди та алгоритми noise reduction, які дозволяють покращити якість вхідних даних перед формуванням спектральних ознак.

Окремою проблемою для систем автоматичного розпізнавання мовлення є наявність тривалих пауз між словами або реченнями. Надмірна кількість пауз може негативно впливати на стабільність потокового розпізнавання та ускладнювати визначення меж фраз. У режимі реального часу система повинна коректно визначати моменти початку та завершення мовлення, щоб уникати зайвих затримок або передчасного завершення транскрипції. Саме тому у сучасних ASR-системах використовуються механізми voice activity detection, які дозволяють відокремлювати мовлення від тиші.

Довжина аудіофрагмента також безпосередньо впливає на якість розпізнавання мовлення. Занадто короткі фрагменти можуть не містити достатнього контексту для правильного прогнозування слів, тоді як надто довгі аудіопослідовності збільшують навантаження на модель та підвищують затримку обробки. У потокових системах розпізнавання мовлення аудіо зазвичай подається частинами фіксованої довжини, що дозволяє забезпечити баланс між швидкістю роботи та точністю транскрипції. Для on-device систем такий баланс є особливо важливим через обмежені ресурси мобільних пристроїв та необхідність роботи у реальному часі.

2.3 Архітектурні принципи сучасних моделей розпізнавання мовлення

Сучасні системи автоматичного розпізнавання мовлення являють собою складні програмно-алгоритмічні комплекси, які виконують послідовне перетворення мовленнєвого сигналу у текстове представлення. Основною задачею таких систем є аналіз акустичних характеристик мовлення, виділення інформативних ознак та прогнозування найбільш ймовірної текстової послідовності.

Загальна структура типової ASR-системи (рисунок 2.9) включає модулі попередньої обробки сигналу, акустичну модель, мовну модель та декодер, який формує фінальний результат розпізнавання.

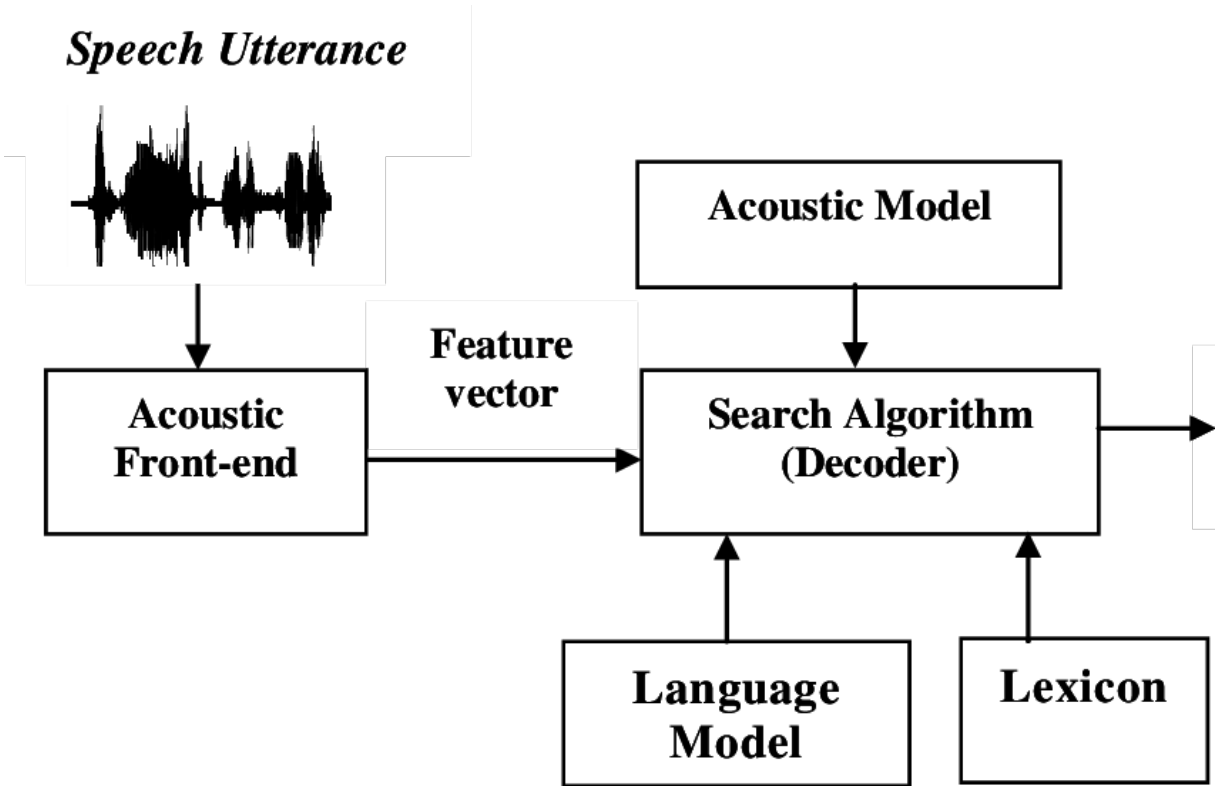


Рисунок 2.9 – Загальна архітектура системи автоматичного розпізнавання мовлення

У сучасних моделях розпізнавання мовлення особливу роль відіграють нейронні мережі, які дозволяють автоматично виділяти складні закономірності у мовленнєвому сигналі. На відміну від класичних статистичних підходів, сучасні глибокі архітектури здатні працювати з великими обсягами даних та ефективно враховувати контекст мовлення. Найбільш поширеними архітектурами у задачах ASR є рекурентні нейронні мережі, encoder-decoder моделі та трансформери, які забезпечують високу точність розпізнавання навіть у складних акустичних умовах.

Важливим компонентом сучасних ASR-систем є механізм декодування, який виконує пошук найбільш ймовірної текстової

послідовності на основі вихідних даних акустичної моделі та мовної моделі. Акустична модель відповідає за аналіз аудіосигналу та визначення ймовірностей окремих звуків або токенів, тоді як мовна модель дозволяє враховувати граматичні та статистичні закономірності природної мови. Комбінування цих компонентів забезпечує більш точне та стабільне розпізнавання мовлення у реальних умовах використання.

У сучасних on-device системах розпізнавання мовлення архітектура моделей додатково оптимізується з урахуванням обмежених ресурсів мобільних пристроїв. Це стосується зменшення розміру моделей, оптимізації обчислень та використання апаратного прискорення через GPU або Neural Engine. Саме завдяки розвитку ефективних трансформерних архітектур та спеціалізованих фреймворків типу Core ML стало можливим виконання складних моделей автоматичного розпізнавання мовлення безпосередньо на мобільних пристроях у режимі реального часу.

2.3.1 Кодування аудіопослідовності у нейромережевих моделях

Кодування аудіопослідовності є одним із ключових етапів роботи сучасних систем автоматичного розпізнавання мовлення. Основною задачею encoder-компонента є перетворення вхідного мовленнєвого сигналу у компактне внутрішнє представлення, яке містить найбільш важливу інформацію про акустичну структуру мовлення. Як вхідні дані зазвичай використовуються мел-спектрограми або лог-мел-спектрограми, що формуються на етапі попередньої обробки аудіосигналу.

У процесі кодування нейронна мережа аналізує часово-частотні характеристики мовлення та виділяє ознаки, які дозволяють розрізняти окремі фонеми, склади та слова. У ранніх системах ASR для цього використовувались рекурентні нейронні мережі та LSTM-архітектури, здатні враховувати часову залежність між елементами послідовності. Проте сучасні моделі дедалі частіше використовують трансформерні encoder-

блоки, які забезпечують більш ефективний аналіз довгих аудіопослідовностей та краще масштабуються при роботі з великими обсягами даних.

Особливістю encoder-компонента є формування прихованих векторних представлень, які зберігають інформацію про мовленнєвий контекст. Такі представлення використовуються декодером для прогнозування текстової послідовності. Якість кодування безпосередньо впливає на точність розпізнавання мовлення, оскільки недостатньо інформативні або спотворені ознаки можуть призводити до помилкового прогнозування слів та фонем.

У сучасних on-device системах кодування аудіопослідовностей повинно виконуватись із мінімальними затримками та помірним використанням ресурсів пристрою. Саме тому архітектури encoder-моделей оптимізуються шляхом зменшення кількості параметрів, використання квантування та застосування апаратного прискорення через GPU або Neural Engine мобільного пристрою.

2.3.2 Декодування тексту за акустичними ознаками

Декодування є етапом, на якому внутрішні представлення, сформовані encoder-компонентом, перетворюються у текстову послідовність. Основною задачею decoder-модуля є прогнозування найбільш ймовірного набору символів, токенів або слів на основі акустичних ознак, отриманих під час аналізу мовлення. Саме на цьому етапі система формує фінальний результат автоматичного розпізнавання мовлення.

У сучасних моделях ASR декодування виконується послідовно, коли кожен наступний токен прогнозується з урахуванням попереднього контексту. Такий підхід дозволяє враховувати граматичні закономірності мови та підвищувати точність формування тексту. У трансформерних

моделях decoder-компонент використовує механізм autoregressive generation, за якого текст генерується поетапно від початку до завершення фрази.

Важливу роль під час декодування відіграє мовна модель, яка дозволяє враховувати статистичні залежності між словами та реченнями. Навіть якщо акустичні ознаки містять неоднозначності або спотворення, мовна модель допомагає вибрати найбільш логічний варіант тексту. Завдяки цьому сучасні системи ASR здатні коректно працювати навіть у шумному середовищі або при нестабільній вимові користувача.

У потокових системах розпізнавання мовлення декодування повинно виконуватись у режимі реального часу, що створює додаткові вимоги до швидкодії алгоритмів. Система повинна формувати проміжні результати транскрипції ще до завершення мовлення користувача, постійно оновлюючи текст залежно від нового контексту. Це особливо важливо для мобільних застосунків, які орієнтовані на інтерактивну голосову взаємодію.

2.3.3 Механізм уваги у задачах перетворення мовлення в текст

Механізм уваги є одним із найважливіших компонентів сучасних трансформерних моделей автоматичного розпізнавання мовлення. Його основною задачею є визначення того, які частини вхідної аудіопослідовності є найбільш важливими для прогнозування конкретного фрагмента тексту. На відміну від класичних рекурентних мереж, механізм уваги дозволяє аналізувати всю послідовність одночасно та враховувати довгострокові залежності між елементами сигналу.

У задачах перетворення мовлення в текст attention-механізм дозволяє decoder-компоненту звертатися до різних ділянок encoder-представлення залежно від поточного етапу генерації тексту. Завдяки цьому модель може ефективно встановлювати відповідність між акустичними ознаками та окремими словами або фонемами. Такий підхід суттєво підвищує точність

розпізнавання мовлення, особливо для довгих речень та складних мовленнєвих конструкцій.

Особливістю трансформерних архітектур є використання self-attention механізму, за якого кожен елемент послідовності аналізується у взаємозв’язку з усіма іншими елементами. Це дозволяє моделі враховувати контекст не лише локально, а й на великій часовій дистанції. У результаті система краще розуміє структуру мовлення, інтонаційні особливості та зв’язки між словами у реченні.

У сучасних моделях типу Whisper використовується multi-head attention, який дозволяє одночасно аналізувати різні аспекти мовленнєвого сигналу. Кожна «голова» attention-механізму може фокусуватись на окремих характеристиках аудіо, наприклад фонетичних особливостях, ритмі мовлення або контекстних залежностях між словами. Саме використання attention-механізмів стало одним із ключових факторів розвитку високоточних систем автоматичного розпізнавання мовлення нового покоління.

2.3.4 Трансформерна архітектура в системах ASR

Трансформерна архітектура стала одним із найважливіших етапів розвитку сучасних систем автоматичного розпізнавання мовлення. На відміну від рекурентних нейронних мереж, трансформери дозволяють обробляти елементи послідовності паралельно, що суттєво підвищує швидкість навчання та ефективність роботи моделей. Саме трансформерний підхід лежить в основі більшості сучасних високоточних ASR-систем, зокрема моделей типу Whisper.

Загальна структура encoder-decoder трансформера для задач розпізнавання мовлення (рисунок 2.10) демонструє взаємодію компонентів кодування, механізмів уваги та декодування текстової послідовності.

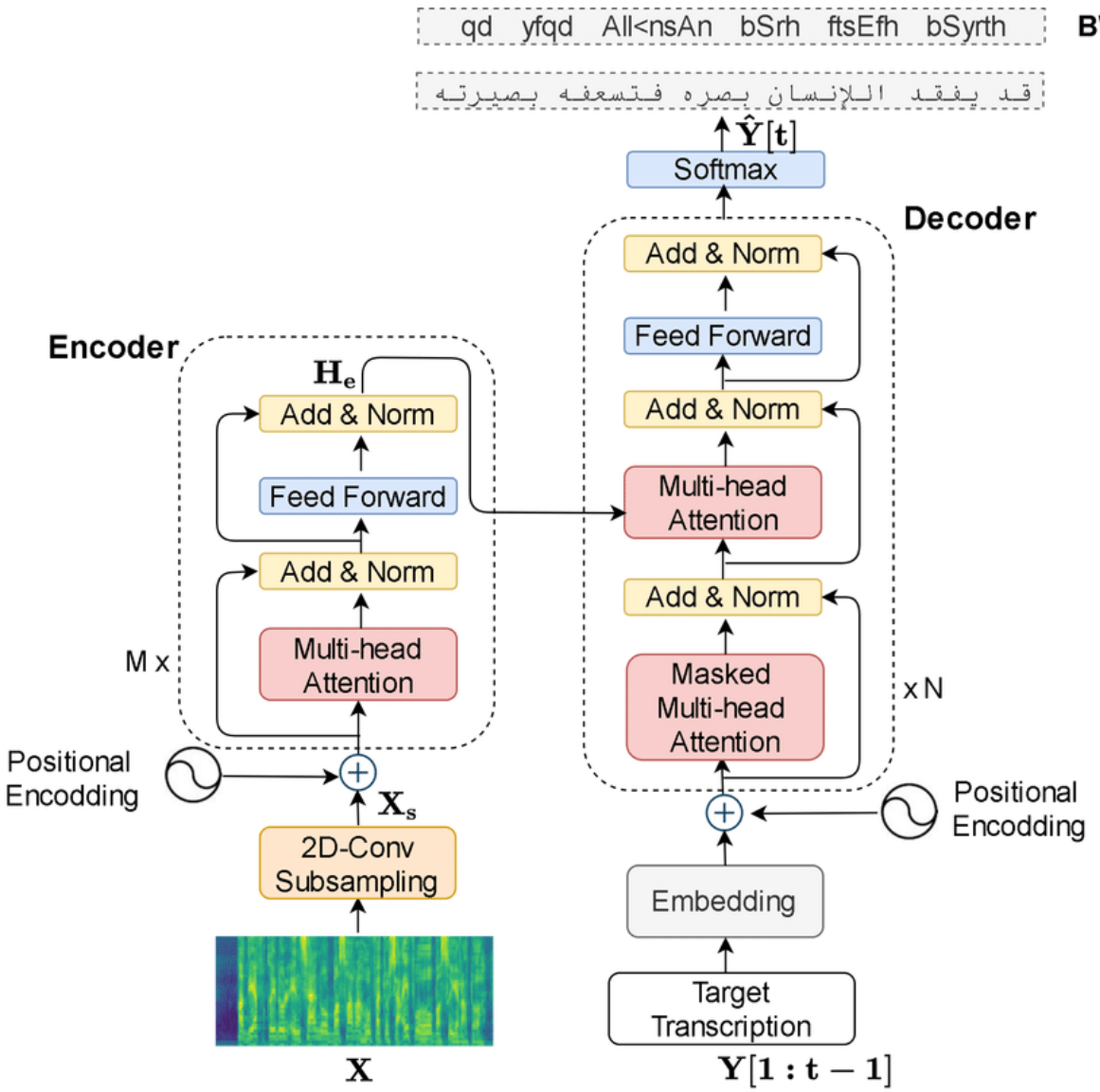


Рисунок 2.10 – Трансформерна encoder-decoder архітектура системи розпізнавання мовлення

Encoder-компонент трансформера відповідає за аналіз вхідного аудіосигналу та формування прихованих представлень мовлення. Перед подачею в encoder аудіосигнал перетворюється у лог-мел-спектрограму, після чого над отриманими ознаками виконуються операції attention-аналізу та feed-forward обробки. Для врахування порядку елементів у послідовності використовується positional encoding, який додає інформацію про часове розташування фрагментів сигналу. Це дозволяє моделі ефективно

працювати з часовими залежностями мовлення без використання рекурентних механізмів.

Decoder-компонент трансформера виконує послідовну генерацію текстового результату на основі прихованих представлень, сформованих encoder-модулем. Під час кожного кроку декодування модель аналізує вже згенеровані токени та визначає найбільш ймовірний наступний елемент тексту. Для цього використовується masked multi-head attention, який дозволяє враховувати лише попередні токени без доступу до майбутнього контексту. Такий підхід забезпечує autoregressive генерацію тексту та дозволяє формувати граматично узгоджені результати транскрипції.

Однією з ключових переваг трансформерних архітектур є використання multi-head attention механізму, який дозволяє моделі одночасно аналізувати різні аспекти мовленнєвого сигналу. Завдяки цьому система може враховувати як локальні акустичні особливості, так і довгострокові залежності між словами та фразами. Це особливо важливо для задач розпізнавання природного мовлення, де значення окремих слів часто залежить від контексту всього речення.

Трансформерні моделі демонструють високу ефективність у задачах ASR, однак одночасно характеризуються значними обчислювальними витратами через велику кількість параметрів та великою складністю attention-обчислень. Саме тому для on-device використання такі моделі потребують оптимізації, квантування та адаптації до можливостей мобільних пристроїв. Розвиток Core ML, Neural Engine та оптимізованих трансформерних архітектур дозволив виконувати подібні моделі безпосередньо на iOS-пристроях у режимі реального часу.

2.4 Теоретичні особливості моделі Whisper

Модель Whisper є однією з найсучасніших трансформерних систем автоматичного розпізнавання мовлення, розробленою компанією OpenAI

для задач багатомовного speech-to-text перетворення. Особливістю цієї моделі є використання encoder-decoder трансформерної архітектури, здатної ефективно працювати з великими обсягами аудіоданих та виконувати розпізнавання мовлення у різних мовах і акустичних умовах. На відміну від багатьох попередніх ASR-систем, Whisper була навчена на масштабному наборі реальних аудіозаписів, що дозволило підвищити стійкість моделі до шумів, різних акцентів, швидкого темпу мовлення та інших факторів, характерних для реального середовища використання.

Однією з ключових особливостей моделі Whisper є те використання лог-мел-спектрограм як вхідного представлення аудіосигналу та застосування attention-механізмів для аналізу часово-частотних залежностей мовлення. Модель виконує автоматичне кодування акустичних ознак та поетапне декодування текстової послідовності, формуючи транскрипцію у режимі autoregressive generation. Завдяки підтримці різних розмірів моделей та можливості оптимізації для локального виконання Whisper стала однією з найбільш перспективних основ для реалізації саме on-device систем розпізнавання мовлення на мобільних пристроях. У межах даного підрозділу розглядаються теоретичні особливості архітектури Whisper, принципи формування лог-мел-спектрограм, механізми кодування та декодування, а також особливості багатомовного розпізнавання мовлення.

2.4.1 Загальна концепція моделі Whisper

Whisper є трансформерною моделлю автоматичного розпізнавання мовлення, побудованою за принципом encoder-decoder архітектури. Основною задачею моделі є перетворення аудіосигналу у текстову послідовність із врахуванням мовного контексту та акустичних особливостей мовлення. Загальна структура моделі Whisper (рисунок 2.11)

включає модулі кодування лог-мел-спектрограми, attention-механізми та decoder-компонент, який виконує генерацію тексту у покроковому режимі.

На етапі попередньої обробки аудіосигнал швидко перетворюється у лог-мел-спектрограму, яка подається на вхід encoder-модуля. Encoder виконує аналіз часово-частотних характеристик мовлення та формує приховані представлення, що містять інформацію про акустичну структуру сигналу. Для врахування часової послідовності фрагментів аудіо використовуються positional encoding механізми, які дозволяють моделі розуміти порядок елементів у вхідній послідовності.

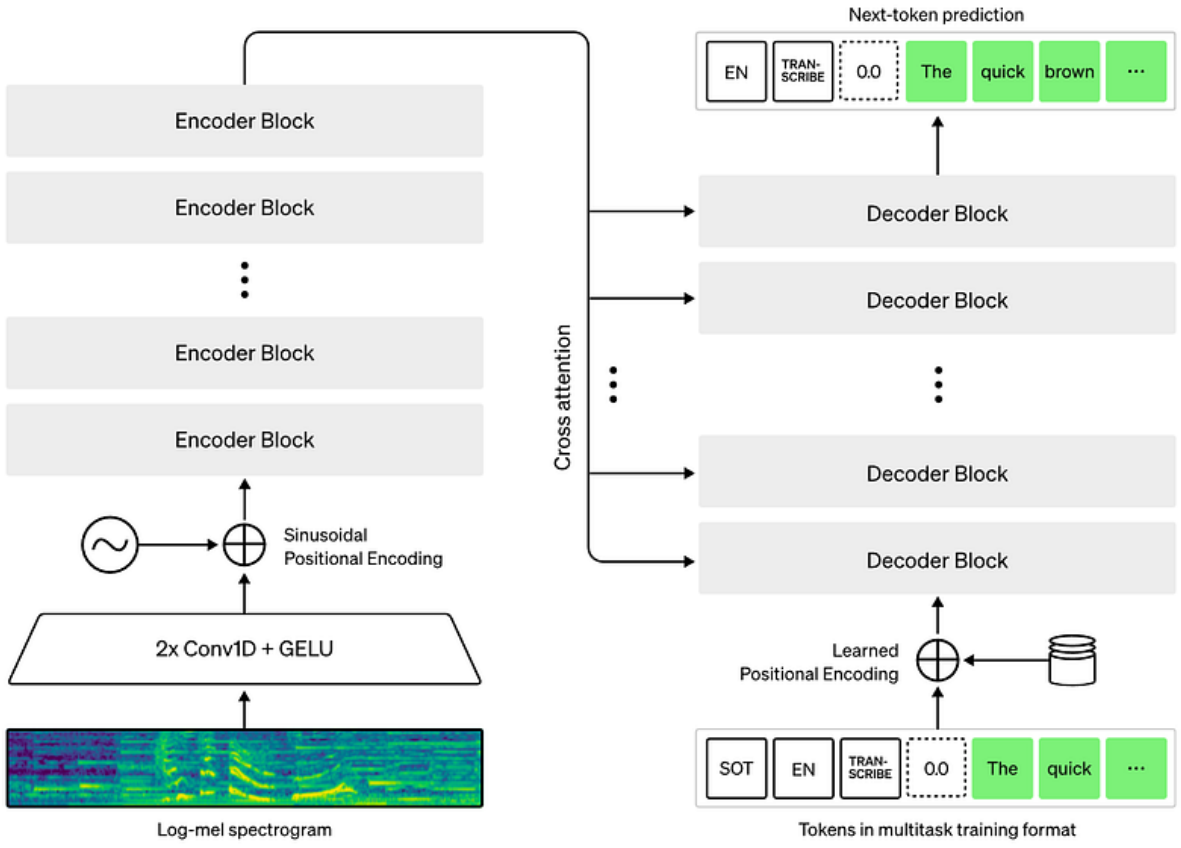


Рисунок 2.11 – Загальна архітектура моделі Whisper

Decoder-компонент моделі Whisper виконує autoregressive генерацію тексту, прогнозуючи наступний токен на основі вже сформованої частини транскрипції та інформації, отриманої від encoder-модуля. Для взаємодії

між encoder та decoder використовується cross-attention механізм, який дозволяє decoder-компоненту звертатися до найбільш релевантних фрагментів аудіопредставлення під час формування тексту. Такий підхід забезпечує високу точність розпізнавання навіть у випадках складного або шумного мовлення.

Однією з важливих особливостей Whisper є підтримка багатомовного розпізнавання мовлення та універсальний формат токенізації. Модель використовує спеціальні службові токени для визначення мови, режиму транскрипції та додаткових параметрів генерації тексту. Завдяки цьому Whisper може працювати з великою кількістю мов без необхідності використання окремих моделей для кожної з них.

Whisper також характеризується високою стійкістю до шумів, різних акцентів та нестандартної вимови. Це стало можливим завдяки навчанню моделі на великому наборі реальних аудіоданих із різноманітними умовами запису. Саме поєднання трансформерної архітектури, attention-механізмів та масштабного навчального набору даних дозволило Whisper стати однією з найефективніших моделей сучасного автоматичного розпізнавання мовлення.

2.4.2 Перетворення аудіосигналу у лог-мел-спектрограму

Одним із ключових етапів роботи моделі Whisper є перетворення початкового аудіосигналу у лог-мел-спектрограму. Саме таке представлення використовується як вхідні дані для encoder-компонента моделі. Процес формування лог-мел-спектрограми (рисунок 2.12) включає кілька послідовних етапів обробки аудіо, починаючи від waveform-сигналу та завершуючи компактним частотно-часовим поданням мовлення.

На першому етапі мовленнєвий сигнал подається у вигляді часової waveform-форми, яка містить зміни амплітуди у часі. Однак безпосереднє використання waveform-сигналу для роботи трансформерної моделі є

малоефективним через велику кількість надлишкової інформації та складність аналізу часових коливань. Саме тому виконується короткочасний спектральний аналіз за допомогою Short-Time Fourier Transform, у результаті якого формується STFT-спектрограма.

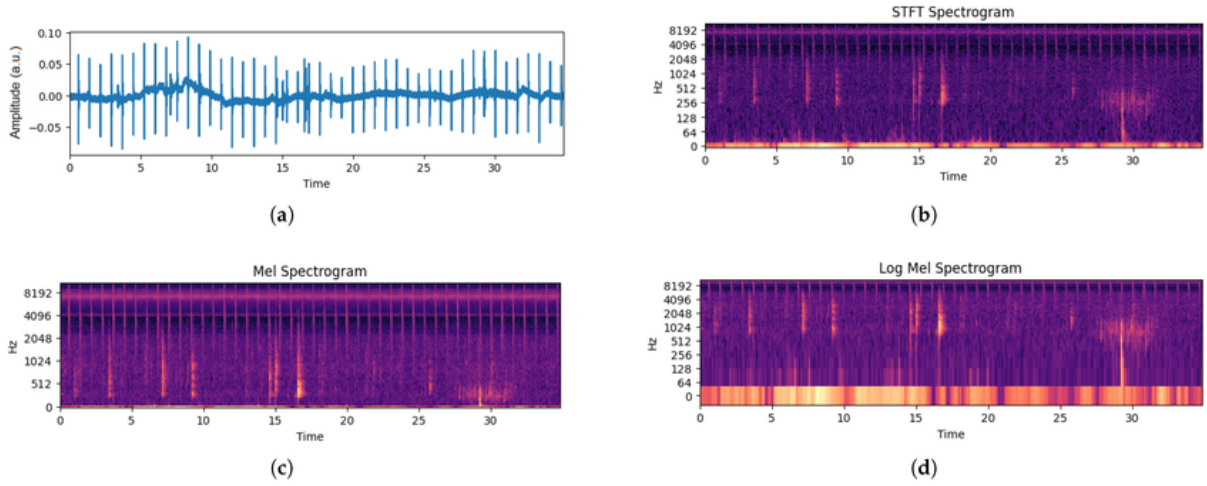


Рисунок 2.12 – Етапи перетворення аудіосигналу у лог-мел-спектрограму

Після формування спектрограми застосовується надійне перетворення у мел-шкалу, яка враховує особливості людського слухового сприйняття. Мел-шкала забезпечує більш детальне представлення низьких частот та менш детальне представлення високочастотних компонент сигналу. У результаті формується мел-спектрограма, яка містить компактне та інформативне подання мовлення. Для додаткового зменшення динамічного діапазону енергії застосовується логарифмічне масштабування, після чого отримується лог-мел-спектрограма.

Лог-мел-спектрограма є значно компактнішою у порівнянні з початковим waveform-сигналом, що дозволяє зменшити обсяг обчислень під час роботи моделі. Саме тому лог-мел-спектрограми стали стандартним способом подання мовлення у більшості сучасних трансформерних ASR-моделей.

2.4.3 Encoder-decoder структура моделі Whisper

Архітектура моделі Whisper побудована за наступним принципом encoder-decoder трансформера, який забезпечує послідовне перетворення аудіосигналу у текстову транскрипцію. Основною задачею такого швидкого encoder-компонента є аналіз лог-мел-спектрограми та формування прихованих представлень мовлення, тоді як decoder-компонент виконує генерацію текстової послідовності. Загальна структура взаємодії encoder та decoder модулів (рисунок 2.13) демонструє процес передачі акустичних ознак між компонентами моделі та подальше формування текстового результату.

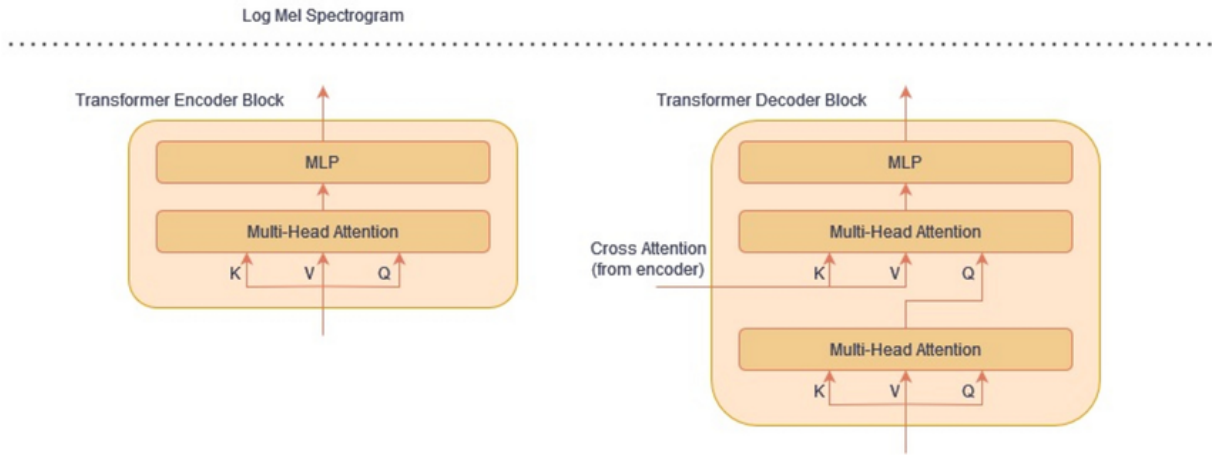


Рисунок 2.13 – Encoder-decoder структура моделі Whisper

Encoder-компонент отримує на вхід лог-мел-спектрограму, над якою попередньо виконуються згорткові операції для зменшення розмірності та виділення локальних акустичних ознак. Після цього сигнал проходить через послідовність transformer encoder блоків, які використовують multi-head attention механізми для аналізу залежностей між різними частинами аудіопослідовності. Для врахування порядку фрагментів мовлення застосовується sinusoidal positional encoding, що дозволяє моделі працювати з часовою структурою сигналу.

Decoder-компонент виконує autoregressive генерацію тексту, прогнозуючи наступний токен на основі вже сформованої частини транскрипції та прихованих представлень encoder-модуля. На кожному кроці decoder використовує masked multi-head attention для аналізу попередніх токенів та cross-attention механізм для взаємодії з encoder представленням аудіосигналу. Саме cross-attention забезпечує зв’язок між акустичними ознаками мовлення та текстовими елементами транскрипції.

Особливістю decoder-компонента є використання learned positional encodings, які дозволяють моделі ефективно враховувати порядок текстових токенів під час генерації результату. На відміну від encoder-модуля, де positional encoding застосовується до аудіопослідовності, decoder працює вже з текстовими токенами та формує мовний контекст для прогнозування наступних слів. Це дозволяє моделі генерувати граматично узгоджений текст та враховувати довгострокові залежності між словами у реченні.

Encoder-decoder архітектура Whisper забезпечує високу точність розпізнавання мовлення навіть у складних акустичних умовах. Завдяки використанню attention-механізмів модель здатна ефективно працювати з довгими аудіопослідовностями, враховувати контекст мовлення та формувати стабільні результати транскрипції. Саме така архітектура стала основою сучасних високоточних систем автоматичного розпізнавання мовлення нового покоління.

2.4.4 Багатомовність, токенізація та формування текстового результату

Однією з ключових особливостей моделі Whisper є підтримка багатомовного автоматичного розпізнавання мовлення. На відміну від багатьох класичних ASR-систем, які орієнтовані лише на одну мову або потребують окремих моделей для різних мов, Whisper використовує єдину універсальну архітектуру для роботи з великою кількістю мовних даних.

Такий підхід став можливим завдяки навчанню моделі на масштабному багатомовному наборі аудіозаписів, що містив різні акценти, темпи мовлення та умови запису. У результаті модель здатна автоматично визначати мову мовлення та виконувати транскрипцію без попереднього налаштування користувачем.

Для представлення текстової інформації Whisper використовує токенізацію, за якої текст розбивається на окремі елементи – токени. Токенами можуть бути окремі символи, частини слів або цілі слова залежно від структури словника моделі. Використання токенізації дозволяє значно зменшити складність задачі генерації тексту та ефективніше працювати з великими мовними словниками. Під час роботи decoder-компонент прогнозує не готові слова, а послідовність токенів, які надалі об’єднуються у завершений текстовий результат.

Особливістю Whisper є використання спеціальних службових токенів, які визначають параметри роботи моделі. До таких токенів належать токени вибору мови, режиму транскрипції, перекладу та службові маркери початку або завершення генерації тексту. Завдяки цьому модель може працювати у різних режимах без зміни архітектури або окремого перенавчання. Наприклад, Whisper здатна не лише виконувати транскрипцію мовлення, а й автоматично перекладати аудіо англійською мовою.

Формування текстового результату у Whisper виконується у режимі autoregressive generation, коли кожен наступний токен прогнозується на основі попередньо сформованої частини тексту. Такий підхід дозволяє враховувати мовний контекст та граматичні закономірності природної мови. У процесі генерації decoder-компонент постійно аналізує як уже сформовані токени, так і приховані представлення аудіосигналу, отримані від encoder-модуля. Саме це забезпечує узгодженість та природність фінальної транскрипції.

У задачах on-device розпізнавання мовлення багатомовність та токенізація створюють додаткові вимоги до продуктивності системи,

оскільки модель повинна працювати з великим словником токенів та підтримувати декілька мов одночасно. Саме тому під час інтеграції Whisper у мобільні застосунки особливу увагу приділяють оптимізації процесу декодування, скороченню затримок генерації тексту та ефективному використанню ресурсів мобільного пристрою.

2.5 Потокове розпізнавання мовлення в реальному часі

Потокове розпізнавання мовлення є одним із найважливіших напрямів розвитку сучасних ASR-систем, особливо у сфері мобільних застосунків та голосових інтерфейсів. На відміну від класичного offline-розпізнавання, де весь аудіозапис аналізується лише після завершення запису, потоковий підхід дозволяє формувати текстову транскрипцію безпосередньо під час мовлення користувача. Це забезпечує швидку взаємодію із системою та створює ефект миттєвого відображення результату.

Реалізація потокового розпізнавання потребує вирішення низки складних задач, пов’язаних із затримками обробки, нестачею контексту та нестабільністю проміжних результатів транскрипції. Система повинна одночасно підтримувати високу швидкість роботи, мінімальне енергоспоживання та достатню точність розпізнавання мовлення. Особливо важливими ці аспекти є для on-device систем, які виконують усі обчислення безпосередньо на мобільному пристрої в режимі реального часу.

2.5.1 Принцип обробки аудіопотоку фрагментами

У потокових системах розпізнавання мовлення аудіосигнал не обробляється як єдиний завершений запис, а подається до моделі невеликими фрагментами фіксованої тривалості. Такий підхід дозволяє виконувати аналіз мовлення поступово та формувати текстову транскрипцію ще до завершення фрази користувача. Найчастіше довжина

одного аудіофрагмента становить від декількох сотень мілісекунд до кількох секунд залежно від архітектури системи та вимог до затримки.

Після надходження нового фрагмента виконується його попередня обробка, формування лог-мел-спектрограми та передача даних до моделі розпізнавання мовлення. Отримані результати об’єднуються з попередньою транскрипцією та постійно оновлюються у процесі мовлення користувача.

2.5.2 Баланс між затримкою розпізнавання та точністю

Однією з головних проблем потокового розпізнавання мовлення є необхідність забезпечення балансу між швидкістю роботи системи та точністю транскрипції. Зменшення розміру аудіофрагментів дозволяє скоротити затримку появи тексту, однак водночас модель отримує менше контекстної інформації для правильного прогнозування слів. Це може призводити до збільшення кількості помилок під час розпізнавання мовлення.

Збільшення тривалості аудіофрагментів, навпаки, дозволяє покращити точність роботи системи завдяки більшому контексту, однак підвищує затримку формування текстового результату. У задачах реального часу особливо важливо знайти компроміс між швидкістю оновлення транскрипції та якістю розпізнавання.

2.5.3 Проблема контексту під час інкрементального формування тексту

Під час потокового розпізнавання система формує текст поступово, отримуючи лише частину мовленнєвого сигналу у кожен момент часу. Через це модель може не мати достатнього контексту для правильного визначення окремих слів або завершення фраз. Особливо це проявляється у

випадках омонімів, складних мовних конструкцій або швидкого мовлення користувача.

У процесі надходження нових аудіоданих система може змінювати вже сформовані частини транскрипції, уточнюючи попередні результати залежно від нового контексту. Такий механізм дозволяє підвищити точність фінального тексту, однак створює проблему нестабільності проміжної транскрипції. Саме тому сучасні ASR-моделі використовують attention-механізми та мовні моделі для ефективнішого врахування довгострокових залежностей між словами.

2.5.4 Стабілізація проміжних результатів транскрипції

У потокових системах автоматичного розпізнавання мовлення проміжні результати транскрипції можуть постійно змінюватися у процесі надходження нових аудіофрагментів. Це пов’язано з тим, що модель отримує додатковий контекст та коригує раніше сформовані слова або фрази. Часті зміни тексту можуть негативно впливати на користувацький досвід та ускладнювати сприйняття транскрипції у режимі реального часу.

Для зменшення нестабільності сучасні системи використовують механізми стабілізації тексту, які дозволяють фіксувати окремі частини транскрипції після досягнення достатнього рівня впевненості моделі. Також можуть застосовуватись порогові значення confidence score, часові затримки оновлення тексту або алгоритми згладжування результатів.

2.6 Теоретичні аспекти оптимізації моделей для мобільних пристроїв

Сучасні моделі автоматичного розпізнавання мовлення характеризуються великою кількістю параметрів та значними обчислювальними витратами, що створює складнощі під час їх використання на мобільних пристроях. На відміну від серверних систем,

смартфони та планшети мають обмежені ресурси процесора, оперативної пам’яті та енергоспоживання, тому пряме використання великих нейронних моделей без оптимізації часто є неможливим. Саме тому одним із ключових напрямів розвитку on-device систем розпізнавання мовлення є адаптація моделей машинного навчання до умов роботи мобільних платформ.

Оптимізація моделей для мобільних пристроїв охоплює комплекс методів, спрямованих на зменшення розміру моделей, прискорення інференсу та зниження навантаження на апаратні ресурси. До найбільш поширених підходів належать квантування ваг, pruning, зменшення розрядності обчислень, використання спеціалізованих форматів моделей та апаратного прискорення через GPU або Neural Engine. Особливо важливими такі оптимізації є для систем потокового розпізнавання мовлення, які повинні виконувати складні обчислення у режимі реального часу без суттєвого впливу на автономність мобільного пристрою.

У задачах on-device ASR оптимізація моделей безпосередньо впливає не лише на швидкість роботи системи, а й на якість користувацького досвіду. Надмірне навантаження на процесор може призводити до збільшення затримок розпізнавання, перегріву пристрою та швидкого розрядження акумулятора. Водночас надто агресивна оптимізація здатна погіршувати точність розпізнавання мовлення через втрату частини інформації або зменшення складності моделі. Саме тому сучасні мобільні системи використовують компромісні підходи, які дозволяють забезпечити баланс між продуктивністю та точністю роботи нейронної мережі.

У межах даного підрозділу розглядаються основні теоретичні аспекти оптимізації моделей машинного навчання для мобільних пристроїв, обмеження апаратних ресурсів iOS-платформи, методи квантування та зменшення розміру моделей, а також використання апаратного прискорення для ефективного виконання систем автоматичного розпізнавання мовлення у режимі реального часу.

2.6.1 Обмеження обчислювальних ресурсів iOS-пристроїв

Мобільні пристрої мають суттєво обмежені обчислювальні ресурси у порівнянні з серверними або настільними системами, що створює додаткові складнощі під час виконання моделей машинного навчання. У задачах автоматичного розпізнавання мовлення це особливо важливо через необхідність обробки великих обсягів аудіоданих у режимі реального часу. Навіть сучасні iPhone та iPad мають обмеження щодо продуктивності CPU, обсягу оперативної пам’яті та енергоспоживання, тому використання великих ASR-моделей без оптимізації може призводити до значного навантаження на систему.

Додатковою проблемою є необхідність підтримки стабільної роботи застосунку без перегріву пристрою та швидкого розрядження акумулятора. Під час виконання трансформерних моделей значна кількість attention-обчислень та операцій множення матриць створює високе навантаження на процесор та графічну підсистему. Саме тому on-device системи розпізнавання мовлення повинні використовувати оптимізовані архітектури моделей та спеціалізовані механізми апаратного прискорення для забезпечення прийнятної продуктивності на мобільних пристроях.

2.6.2 Квантування та зменшення розміру моделей машинного навчання

Одним із найбільш поширених методів оптимізації моделей машинного навчання є квантування, яке полягає у зменшенні розрядності числових значень ваг та обчислень. У стандартному режимі нейронні мережі часто використовують 32-бітні числа з плаваючою комою, однак для мобільних систем можуть застосовуватись 16-бітні або 8-бітні представлення. Це дозволяє суттєво зменшити обсяг пам’яті, необхідний для зберігання моделі, а також прискорити виконання обчислень.

Крім квантування, для зменшення розміру моделей використовуються pruning та knowledge distillation підходи. Pruning дозволяє видаляти малозначущі параметри нейронної мережі без суттєвого погіршення якості роботи моделі, тоді як knowledge distillation передбачає перенесення знань великої моделі у компактнішу архітектуру. Використання таких методів особливо важливе для on-device систем, оскільки дозволяє запускати складні ASR-моделі на мобільних пристроях із обмеженими ресурсами.

2.6.3 Використання апаратного прискорення для виконання нейронних моделей

Для підвищення продуктивності моделей машинного навчання сучасні iOS-пристрої використовують спеціалізовані апаратні компоненти, призначені для виконання нейронних обчислень. До таких компонентів належать GPU та Apple Neural Engine, які здатні значно ефективніше виконувати операції множення матриць та attention-обчислення у порівнянні зі звичайним CPU. Саме використання апаратного прискорення зробило можливим запуск складних моделей автоматичного розпізнавання мовлення безпосередньо на мобільних пристроях.

Фреймворк Core ML автоматично визначає оптимальний спосіб виконання моделі та може розподіляти обчислення між CPU, GPU та Neural Engine залежно від типу операцій і доступних ресурсів. Це дозволяє суттєво скоротити час інференсу та зменшити енергоспоживання системи.

2.6.4 Вплив оптимізації на швидкодію та точність

Оптимізація моделей машинного навчання безпосередньо впливає на швидкодію систем автоматичного розпізнавання мовлення. Зменшення розміру моделі та використання квантування дозволяють скоротити час інференсу, знизити затримку формування транскрипції та покращити

роботу системи у режимі реального часу. Для мобільних застосунків це є критично важливим, оскільки користувач очікує швидкого та стабільного оновлення тексту під час мовлення.

Водночас надмірна оптимізація може негативно впливати на точність розпізнавання мовлення. Зменшення кількості параметрів або використання низькорозрядних представлень іноді призводить до втрати частини інформації та погіршення якості прогнозування тексту. Саме тому під час адаптації моделей для on-device використання необхідно знаходити компроміс між швидкістю роботи системи та точністю транскрипції. Сучасні мобільні ASR-системи використовують комбіновані підходи оптимізації, які дозволяють забезпечити прийнятну якість розпізнавання при обмежених ресурсах пристрою.

2.7 Інтеграція моделей машинного навчання у Core ML

Core ML є основним фреймворком екосистеми iOS для інтеграції та виконання моделей машинного навчання безпосередньо на мобільних пристроях. Основною задачею Core ML є забезпечення ефективного та оптимізованого інференсу нейронних мереж із використанням апаратних можливостей пристрою. Функціональні можливості Core ML (рисунок 2.14) охоплюють задачі класифікації зображень, обробки тексту, розпізнавання мовлення, аналізу аудіосигналів та інших напрямів застосування штучного інтелекту.

Однією з ключових переваг Core ML є підтримка on-device виконання моделей, за якого всі обчислення здійснюються локально без передачі даних до хмарних сервісів. Такий підхід дозволяє суттєво покращити конфіденційність користувацьких даних, зменшити затримки обробки та забезпечити роботу системи навіть за відсутності інтернет-з’єднання.

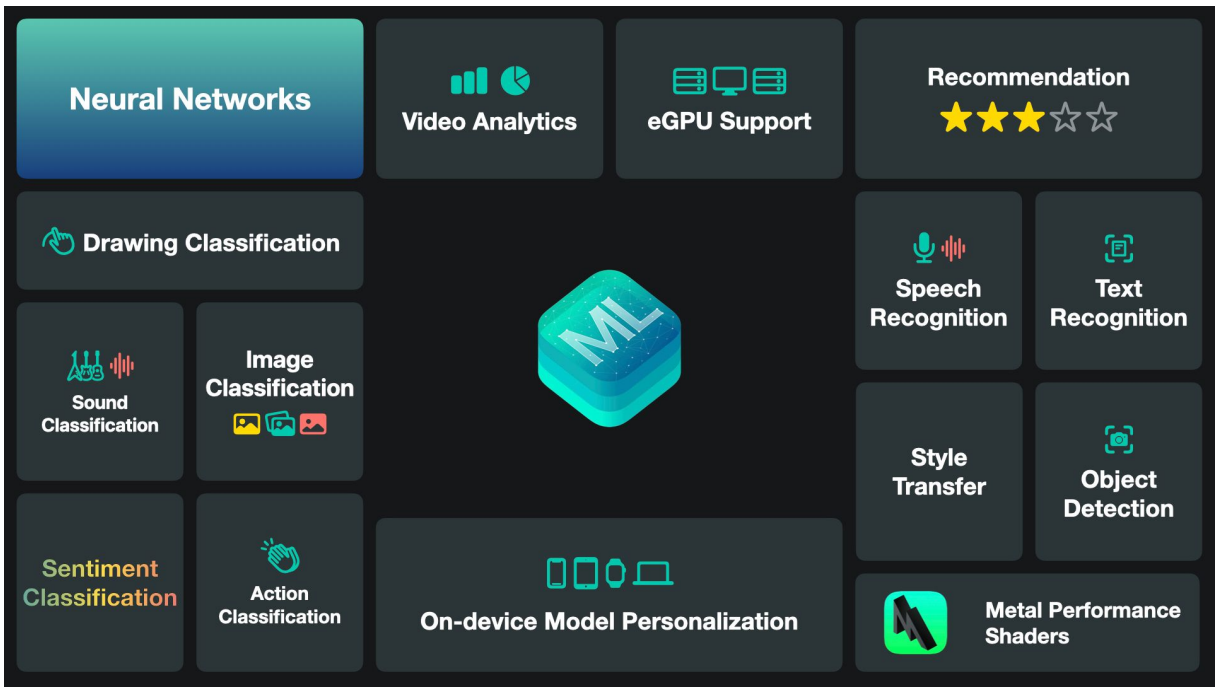


Рисунок 2.14 – Можливості фреймворку Core ML для виконання моделей машинного навчання

Core ML підтримує інтеграцію моделей, створених у різних фреймворках машинного навчання, зокрема TensorFlow, PyTorch та scikit-learn. Перед використанням у iOS-застосунку модель конвертується у спеціальний формат .mlmodel, оптимізований для виконання на пристроях Apple. Під час запуску Core ML автоматично визначає найбільш ефективний спосіб виконання моделі, розподіляючи обчислення між CPU, GPU та Neural Engine залежно від типу операцій та характеристик пристрою.

Важливою особливістю Core ML є підтримка оптимізації моделей для мобільного середовища. Фреймворк дозволяє використовувати квантування, зменшення точності обчислень та інші методи оптимізації для скорочення затримок і зниження енергоспоживання. Завдяки цьому навіть складні трансформерні моделі можуть виконуватись на сучасних iOS-пристроях у режимі реального часу.

У задачах on-device автоматичного розпізнавання мовлення Core ML виступає основою для інтеграції моделей типу Whisper у мобільний

застосунок. Використання Core ML дозволяє забезпечити локальне виконання нейронної мережі, швидке формування транскрипції та ефективне використання апаратних ресурсів пристрою. Саме поєднання Core ML, Neural Engine та оптимізованих трансформерних моделей стало важливим етапом розвитку сучасних мобільних систем розпізнавання мовлення.

2.7.1 Формат моделей Core ML та особливості їх виконання

Формат .mlmodel є стандартним способом представлення моделей машинного навчання в екосистемі Apple. Саме у такому форматі моделі інтегруються у iOS-застосунки та виконуються за допомогою фреймворку Core ML. Загальна схема формування Core ML моделі (рисунок 2.15) демонструє процес конвертації моделей машинного навчання у спеціалізований формат, оптимізований для виконання на пристроях Apple.

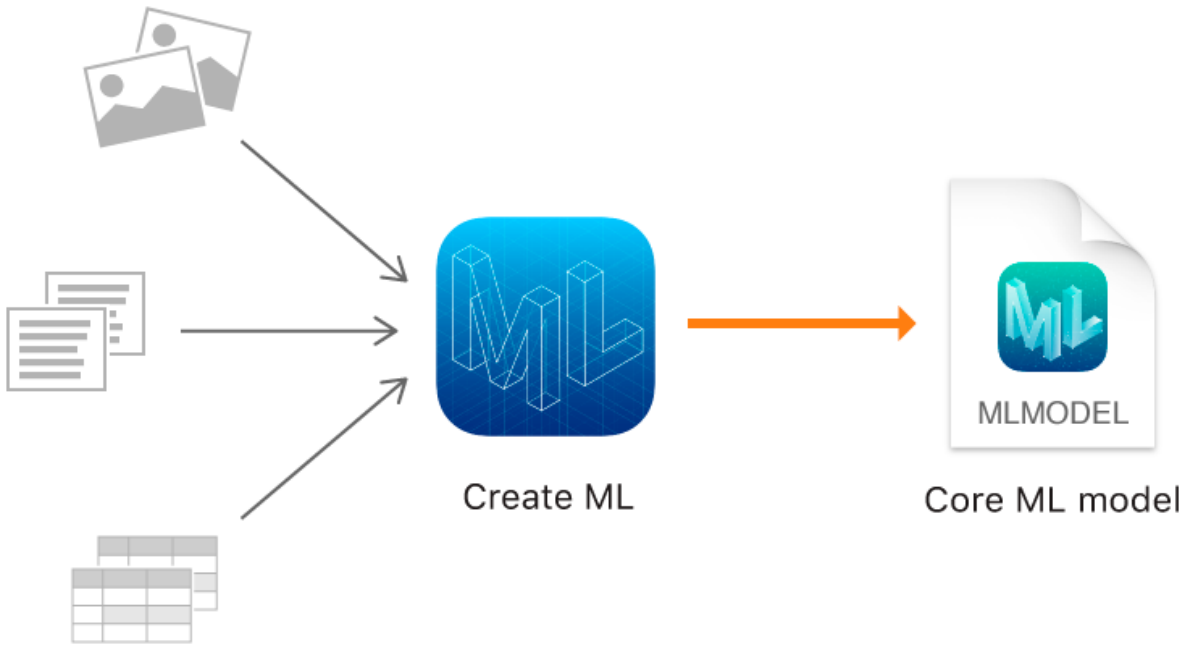


Рисунок 2.15 – Процес формування Core ML моделі у форматі .mlmodel

Формат .mlmodel містить не лише ваги нейронної мережі, а й опис структури моделі, типів вхідних та вихідних даних, параметрів обробки та додаткових метаданих. Завдяки цьому Core ML може автоматично виконувати підготовку даних, оптимізацію обчислень та керування процесом інференсу.

Такий підхід значно спрощує інтеграцію моделей машинного навчання у мобільні застосунки та забезпечує уніфікований механізм роботи з різними типами моделей.

Після інтеграції .mlmodel у застосунок Core ML автоматично компілює модель у внутрішнє оптимізоване представлення, адаптоване до конкретного пристрою. Під час виконання система може використовувати CPU, GPU або Neural Engine залежно від типу обчислень та апаратних можливостей iPhone або iPad. Це дозволяє забезпечити високу швидкість інференсу та ефективне використання ресурсів мобільного пристрою.

Особливістю виконання моделей Core ML є підтримка on-device інференсу без необхідності використання зовнішніх серверів. Усі обчислення виконуються локально, що дозволяє підвищити рівень конфіденційності користувацьких даних та зменшити затримки обробки.

Для задач автоматичного розпізнавання мовлення це є особливо важливим, оскільки система може працювати навіть без доступу до мережі Інтернет та забезпечувати миттєве формування текстової транскрипції.

Core ML також підтримує різні методи оптимізації моделей, включаючи квантування, зменшення точності обчислень та оптимізацію пам’яті.

Це дозволяє адаптувати навіть складні трансформерні архітектури до умов роботи мобільних пристроїв. Саме використання формату .mlmodel та механізмів оптимізованого виконання стало основою розвитку сучасних on-device систем машинного навчання для iOS.

2.7.2 Використання Neural Engine, GPU та CPU для локального інференсу

Під час виконання моделей машинного навчання Core ML може використовувати різні апаратні компоненти iOS-пристрою, зокрема CPU, GPU та Apple Neural Engine. Такий підхід дозволяє ефективно розподіляти обчислювальне навантаження залежно від типу операцій моделі. Взаємодія Core ML із системними компонентами iPhone (рисунок 2.16) демонструє роль апаратного прискорення у задачах комп'ютерного зору, обробки тексту, аналізу аудіо та автоматичного розпізнавання мовлення.

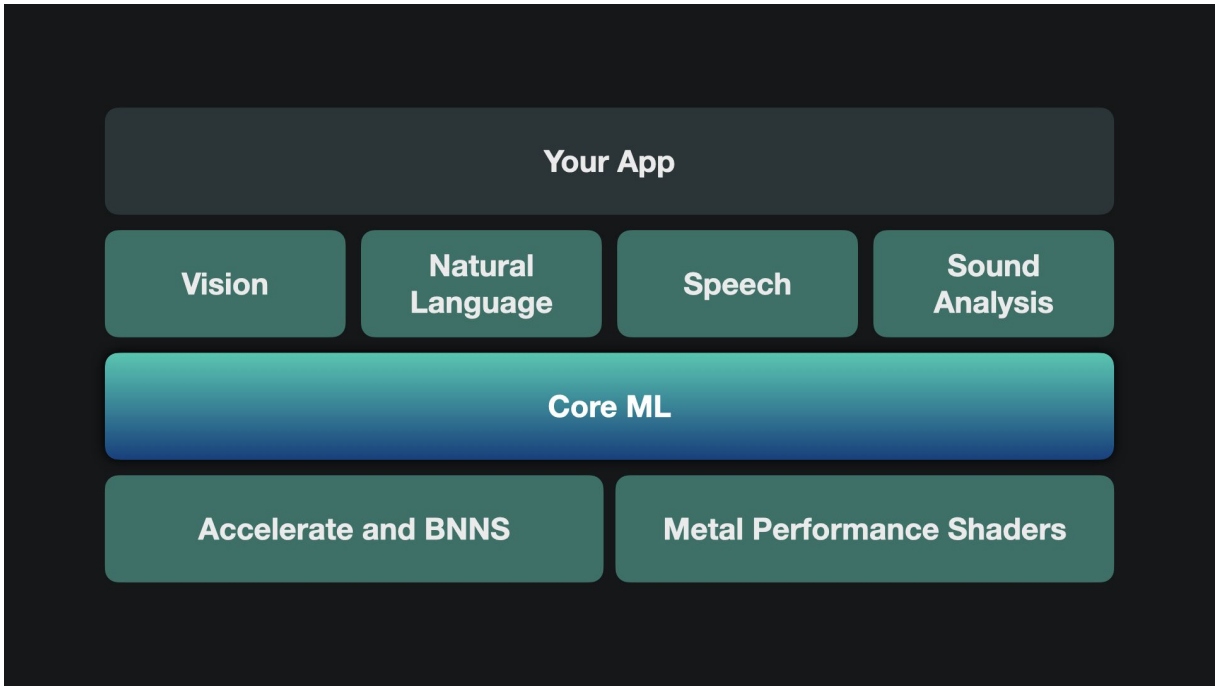


Рисунок 2.16 – Взаємодія Core ML з апаратними компонентами iOS-пристрою

CPU використовується для виконання загальних обчислень, логіки застосунку та операцій, які не потребують масового паралелізму. Хоча центральний процесор здатний виконувати нейронні моделі, його продуктивність у задачах машинного навчання значно нижча у порівнянні з

GPU або Neural Engine. Саме тому використання лише CPU для великих трансформерних моделей може призводити до високих затримок інференсу та збільшення енергоспоживання пристрою.

GPU забезпечує ефективне виконання паралельних математичних операцій, що особливо важливо для нейронних мереж із великою кількістю матричних обчислень. У задачах автоматичного розпізнавання мовлення графічний процесор може використовуватись для прискорення обробки спектрограм, attention-механізмів та інших елементів трансформерної архітектури. Використання GPU дозволяє суттєво скоротити час інференсу та забезпечити стабільнішу роботу системи у режимі реального часу.

Apple Neural Engine є спеціалізованим апаратним модулем, розробленим для прискорення обчислень штучного інтелекту. Neural Engine оптимізований для виконання операцій машинного навчання та здатний виконувати нейронні моделі значно швидше та енергоефективніше у порівнянні з CPU. Саме використання Neural Engine стало одним із ключових факторів розвитку on-device систем машинного навчання в екосистемі Apple.

Core ML автоматично визначає найбільш ефективний спосіб виконання моделі та може динамічно розподіляти обчислення між CPU, GPU та Neural Engine. Такий механізм дозволяє оптимізувати продуктивність системи залежно від типу моделі та доступних ресурсів пристрою. Для задач потокового розпізнавання мовлення це є особливо важливим, оскільки система повинна забезпечувати швидке формування транскрипції при мінімальному енергоспоживанні та стабільній роботі мобільного застосунку.

2.7.3 Обмеження та переваги Core ML у задачах on-device ASR

Core ML забезпечує низку важливих переваг для реалізації on-device систем автоматичного розпізнавання мовлення на iOS-пристроях. Однією з

головних переваг є можливість локального виконання моделей без передачі аудіоданих до зовнішніх серверів. Такий підхід дозволяє підвищити рівень конфіденційності користувача, зменшити затримки обробки та забезпечити роботу системи навіть за відсутності інтернет-з’єднання. Крім того, Core ML автоматично оптимізує виконання моделей із використанням GPU та Neural Engine, що суттєво покращує продуктивність мобільних застосунків.

Ще однією перевагою Core ML є глибока інтеграція з екосистемою Apple та підтримка спеціалізованих механізмів оптимізації моделей машинного навчання. Фреймворк дозволяє використовувати квантування, оптимізацію пам’яті та автоматичне керування ресурсами пристрою, що є особливо важливим для потокового розпізнавання мовлення у режимі реального часу. Водночас Core ML має і певні обмеження, пов’язані з підтримкою окремих типів нейронних операцій та великих трансформерних архітектур. Деякі моделі потребують додаткової адаптації або спрощення перед конвертацією у формат .mlmodel, а надто великі ASR-моделі можуть створювати значне навантаження на пам’ять та апаратні ресурси мобільного пристрою.

2.7.4 Особливості використання Whisper-моделі в iOS

Інтеграція моделі Whisper у iOS-застосунок має низку особливостей, пов’язаних із великою складністю трансформерної архітектури та високими вимогами до обчислювальних ресурсів. Для використання Whisper у середовищі iOS модель зазвичай конвертується у формат Core ML та додатково оптимізується шляхом квантування або зменшення розрядності обчислень. Це дозволяє зменшити розмір моделі та забезпечити її виконання безпосередньо на мобільному пристрої у режимі реального часу. Під час роботи Whisper на iOS особливу увагу приділяють оптимізації потокової обробки аудіо та мінімізації затримок формування транскрипції.

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Формалізація вимог до мобільного застосунку

На етапі проєктування мобільного застосунку важливим завданням є формалізація вимог до майбутньої системи. Чітке визначення функціональних та нефункціональних характеристик дозволяє забезпечити відповідність програмного продукту поставленим задачам, а також сформувати основу для подальшої реалізації архітектури, модулів та механізмів взаємодії компонентів системи. Для мобільних застосунків, що виконують розпізнавання мовлення в реальному часі, особливого значення набувають вимоги до швидкодії, стабільності роботи та ефективності використання апаратних ресурсів мобільного пристрою.

Окрему увагу необхідно приділити вимогам до конфіденційності користувацьких даних, оскільки система працює з аудіосигналами та текстовими транскрипціями, що можуть містити персональну інформацію. У межах розроблюваного застосунку було обрано концепцію on-device обробки, за якої всі основні процеси розпізнавання мовлення та AI-обробки тексту виконуються локально без передачі даних на зовнішні сервери. Такий підхід дозволяє підвищити рівень приватності, зменшити залежність від мережевого з’єднання та забезпечити стабільну роботу системи навіть в автономному режимі.

3.1.1 Функціональні вимоги до системи розпізнавання мовлення

До основних функціональних вимог мобільного застосунку належить можливість запису голосу користувача з мікрофона пристрою та виконання розпізнавання мовлення у реальному часі. Система повинна динамічно відображати текстову транскрипцію під час мовлення користувача без необхідності завершення запису. Після завершення процесу запису

застосунок має формувати фінальну текстову нотатку, яка може бути збережена у локальному сховищі для подальшого перегляду, пошуку або редагування. Важливою функцією також є підтримка паузи, відновлення та скасування запису без втрати поточного стану транскрипції.

Окрім базового розпізнавання мовлення, система повинна підтримувати функції AI-обробки тексту на основі Foundation Models. До таких можливостей належить автоматичне переформатування тексту у вигляді короткого підсумку, списку, електронного листа або структурованих нотаток. Застосунок також повинен забезпечувати перегляд бібліотеки збережених нотаток, пошук за текстом транскрипції та копіювання результатів розпізнавання. Сукупність зазначених функцій формує завершений цикл взаємодії користувача із системою голосових нотаток.

3.1.2 Нефункціональні вимоги до продуктивності та затримки обробки

Однією з ключових нефункціональних вимог до системи є забезпечення мінімальної затримки між моментом вимови фрази користувачем та появою відповідного тексту на екрані. Для забезпечення комфортної взаємодії із застосунком затримка обробки повинна бути достатньо низькою, щоб створювати ефект live-транскрипції в реальному часі. Це вимагає ефективної організації аудіопотоку, оптимізації процесу інференсу нейронної моделі та використання апаратного прискорення мобільного пристрою.

Додатково система повинна забезпечувати стабільну роботу при тривалому записі аудіо без суттєвого зростання навантаження на процесор, пам’ять або акумулятор пристрою. Оскільки мобільний застосунок виконує локальний інференс моделей машинного навчання, важливим є ефективне використання Neural Engine, GPU та CPU iOS-пристрою. Також до нефункціональних вимог належать стабільність роботи інтерфейсу,

відсутність блокування основного потоку виконання та підтримка плавної анімації під час запису й обробки аудіо.

3.1.3 Вимоги до конфіденційності та on-device обробки даних

Розроблюваний мобільний застосунок орієнтований на забезпечення максимального рівня конфіденційності користувацьких даних. У межах реалізованої концепції всі процеси розпізнавання мовлення, збереження транскрипцій та AI-обробки тексту виконуються локально на мобільному пристрої без використання зовнішніх хмарних сервісів. Такий підхід дозволяє уникнути передачі аудіоданих через мережу та мінімізує ризики витоку персональної інформації користувача.

Важливою вимогою також є локальне зберігання нотаток і транскрипцій із використанням SwiftData. Система не повинна автоматично експортувати або синхронізувати дані із зовнішніми серверами без явної дії користувача. Додатково застосунок має коректно обробляти системні дозволи доступу до мікрофона та забезпечувати прозоре інформування користувача щодо використання аудіоданих. Реалізація on-device pipeline позитивно впливає не лише на конфіденційність, але й на стабільність роботи системи при відсутності доступу до Інтернету.

3.1.4 Вимоги до користувацького інтерфейсу

Користувацький інтерфейс мобільного застосунку повинен забезпечувати швидку та інтуїтивно зрозумілу взаємодію з функціями запису, транскрипції та перегляду нотаток. Основний сценарій використання передбачає мінімальну кількість дій для запуску процесу запису голосу, тому інтерфейс має акцентувати увагу на центральних елементах керування записом та відображенні live-транскрипції. Важливою вимогою є адаптація інтерфейсу під сучасні принципи iOS-дизайну з

використанням плавних анімацій, візуальної ієрархії та нативних елементів взаємодії.

Окрему увагу приділено забезпеченню зручного перегляду збережених нотаток та навігації між екранами застосунку. Інтерфейс повинен підтримувати швидкий пошук транскрипцій, відкриття деталей нотатки та виконання AI-перетворення тексту без перевантаження користувача зайвими елементами. Дизайн мого застосунку орієнтований на dark-first концепцію та використання мінімалістичного стилю оформлення, що дозволяє зменшити когнітивне навантаження під час тривалої взаємодії із системою.

3.2 Проєктування функціональної моделі системи

Функціональна модель системи визначає основні програмні модулі мобільного застосунку та взаємозв’язки між ними. Побудова такої моделі дозволяє формалізувати внутрішню логіку роботи системи, визначити послідовність обробки аудіоданих та організувати взаємодію між компонентами, що відповідають за розпізнавання мовлення, обробку тексту та збереження результатів. У межах розроблюваного застосунку функціональна модель (рисунок 3.1) орієнтована на реалізацію повністю локального pipeline обробки мовлення з використанням Core ML.

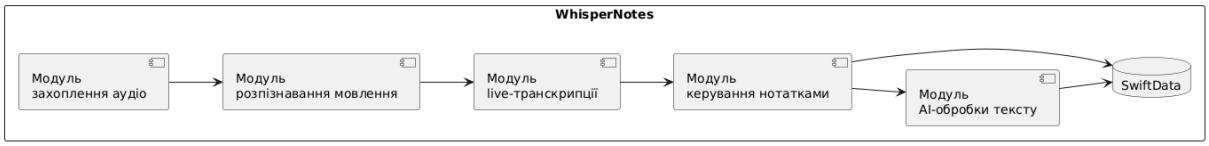


Рисунок 3.1 – Функціональна модель системи

Основу функціональної моделі складає послідовна передача даних між окремими програмними модулями. Спочатку аудіосигнал захоплюється

через модуль роботи з мікрофоном, після чого передається до модуля розпізнавання мовлення на основі Whisper-моделі. Результат розпізнавання надходить до модуля live-транскрипції, який відповідає за динамічне оновлення тексту в інтерфейсі застосунку. Далі сформована нотатка може бути оброблена AI-модулем для виконання трансформації тексту та збережена у локальному сховищі SwiftData (рисунок 3.1).

3.2.1 Модуль захоплення та обробки аудіосигналу

Модуль захоплення та обробки аудіосигналу відповідає за взаємодію застосунку з мікрофоном iOS-пристрою та формування безперервного аудіопотоку для подальшої обробки. У межах роботи модуля здійснюється ініціалізація аудіосесії, отримання PCM-даних у реальному часі, контроль стану запису та підготовка аудіофрагментів до передачі у систему розпізнавання мовлення. Додатково модуль забезпечує підтримку паузи, відновлення та завершення запису без порушення цілісності аудіопотоку.

3.2.2 Модуль розпізнавання мовлення на основі Whisper і Core ML

Модуль розпізнавання мовлення виконує перетворення аудіоданих у текстову транскрипцію з використанням Whisper-моделі, інтегрованої через Core ML. Основним завданням модуля є виконання локального інференсу нейронної мережі в режимі реального часу з мінімальною затримкою обробки.

Для оптимізації продуктивності використовується апаратне прискорення iOS-пристрою, включаючи Neural Engine та GPU, що дозволяє забезпечити стабільну live-транскрипцію без використання хмарних сервісів.

3.2.3 Модуль формування та збереження текстових нотаток

Після завершення процесу розпізнавання мовлення система формує текстову нотатку, яка містить транскрипцію аудіозапису та супровідну інформацію. Модуль відповідає за створення структури нотатки, формування метаданих, збереження тривалості запису та організацію подальшого перегляду тексту. Результати роботи модуля використовуються для відображення бібліотеки нотаток, пошуку транскрипцій та відкриття детального перегляду окремого запису.

3.2.4 Модуль AI-обробки та трансформації транскрипцій

Модуль AI-обробки виконує додаткову трансформацію текстових транскрипцій за допомогою Foundation Models. У межах роботи модуля користувач може перетворювати текст у різні формати, зокрема короткий підсумок, структурований список, електронний лист або нотатки зустрічі. Також підтримується модифікація стилю тексту, включаючи скорочення, підвищення формальності або покращення читабельності результату.

3.2.5 Модуль керування локальними даними

Модуль керування локальними даними забезпечує організацію зберігання нотаток, транскрипцій та результатів AI-обробки із використанням SwiftData. До його функцій належить створення, оновлення, пошук та видалення записів у локальному сховищі застосунку. Використання локальної бази даних дозволяє забезпечити автономну роботу системи, швидкий доступ до інформації та збереження конфіденційності користувацьких даних без передачі їх на зовнішні сервери.

3.3 Моделювання сценаріїв взаємодії користувача із системою

Моделювання сценаріїв взаємодії користувача із системою є важливим етапом проєктування мобільного застосунку, оскільки дозволяє формалізувати логіку роботи інтерфейсу та визначити основні послідовності дій під час використання функціональних можливостей системи. Для застосунків, орієнтованих на роботу в режимі реального часу, особливого значення набуває забезпечення швидкої взаємодії між користувачем та програмними компонентами без перевантаження інтерфейсу зайвими елементами керування.

У межах розроблюваного застосунку було визначено ключові сценарії використання, пов’язані із записом голосу, live-транскрипцією, переглядом нотаток та AI-обробкою тексту. Формалізація таких сценаріїв дозволяє забезпечити логічну структуру навігації між екранами застосунку, оптимізувати взаємодію між функціональними модулями та сформувати основу для подальшого проєктування користувацького інтерфейсу.

3.3.1 Основні користувацькі потоки мобільного застосунку

Основний користувацький потік мобільного застосунку починається з переходу на екран запису голосової нотатки, де користувач отримує можливість запуску процесу захоплення аудіо. Під час мовлення система виконує live-транскрипцію та динамічно відображає розпізнаний текст у режимі реального часу. Після завершення запису користувач може зберегти сформовану нотатку, переглянути її деталі або виконати додаткову AI-обробку тексту для зміни формату чи стилю подання інформації.

Іншим важливим користувацьким потоком є робота з бібліотекою нотаток. Користувач має можливість переглядати список збережених транскрипцій, здійснювати пошук за текстом, відкривати детальний перегляд окремої нотатки та редагувати її вміст. Також підтримуються

сценарії копіювання тексту, видалення нотаток та повторного перегляду AI-перетворених версій транскрипцій. Сукупність зазначених потоків забезпечує повний цикл роботи користувача із системою голосових нотаток.

3.3.2 Сценарій створення голосової нотатки в реальному часі

Сценарій створення голосової нотатки є основним процесом взаємодії користувача із мобільним застосунком. Робота сценарію починається після натискання кнопки запуску запису, у результаті чого ініціалізується модуль захоплення аудіо та формується безперервний аудіопотік. Отримані аудіодані передаються до Whisper-моделі через Core ML, де виконується поетапне розпізнавання мовлення з формуванням часткової транскрипції у режимі реального часу (рисунок 3.2).

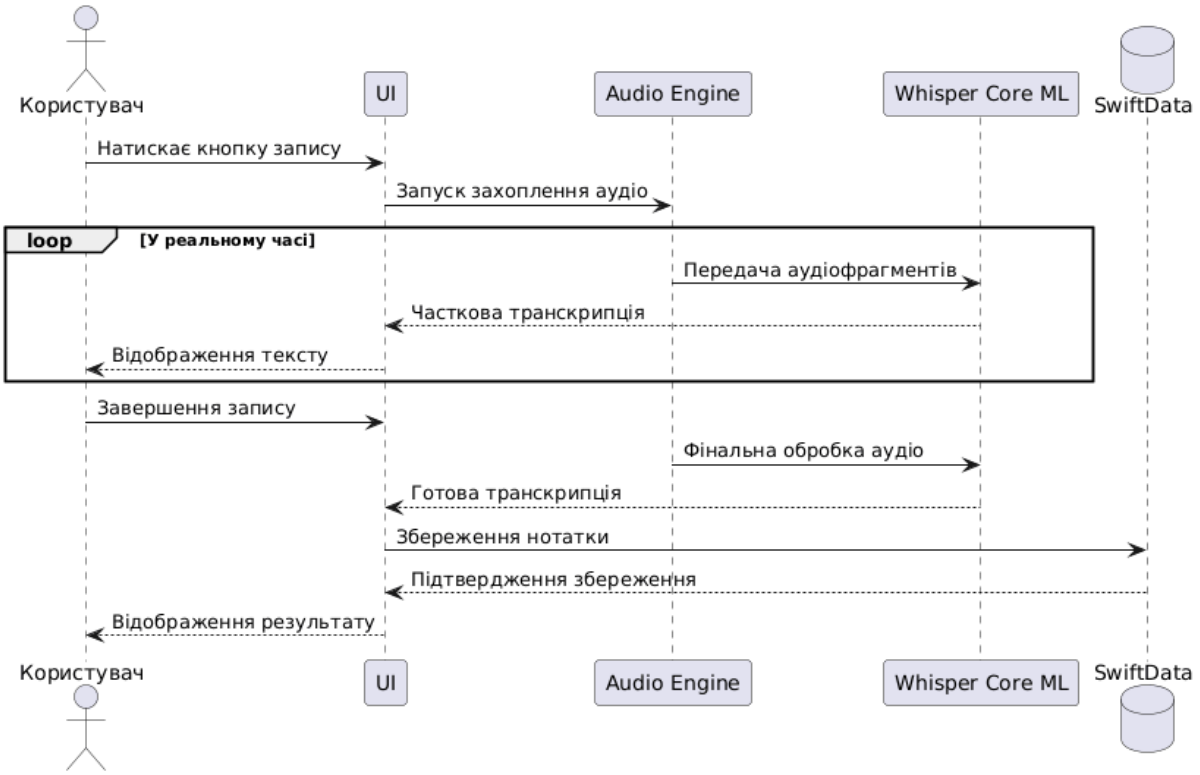


Рисунок 3.2 – Сценарій створення голосової нотатки в реальному часі

Під час запису система постійно оновлює текстову область інтерфейсу, забезпечуючи динамічне відображення результатів розпізнавання мовлення. Після завершення запису виконується фінальна обробка аудіосигналу, формується завершена транскрипція та здійснюється її збереження у локальному сховищі SwiftData. Такий підхід дозволяє забезпечити безперервний процес live-транскрипції без необхідності використання зовнішніх серверів або хмарної обробки даних.

3.3.3 Сценарій перегляду та редагування транскрипції

Сценарій перегляду та редагування транскрипції реалізує механізм роботи користувача із вже збереженими голосовими нотатками. Після вибору необхідної нотатки у бібліотеці застосунок виконує отримання відповідних даних із локального сховища SwiftData, включаючи текст транскрипції та AI-оброблену версію нотатки. Далі відкривається екран детального перегляду, де користувач отримує можливість ознайомитися зі змістом розпізнаного тексту та виконати його редагування (рисунок 3.3).

Після внесення змін система оновлює дані транскрипції у локальному сховищі та відображає оновлений результат в інтерфейсі застосунку. Такий підхід дозволяє користувачу коригувати можливі помилки автоматичного розпізнавання мовлення, уточнювати зміст нотатки та підтримувати актуальність збережених текстових записів. Використання локального зберігання забезпечує швидке оновлення інформації та автономність роботи системи без необхідності мережевої взаємодії.

Додатково сценарій підтримує подальшу AI-обробку відредагованої транскрипції, що дозволяє використовувати оновлений текст як основу для генерації структурованих нотаток, підсумків або інших форматів представлення інформації. Це забезпечує гнучкість роботи із збереженим контентом та розширює можливості повторного використання нотаток.

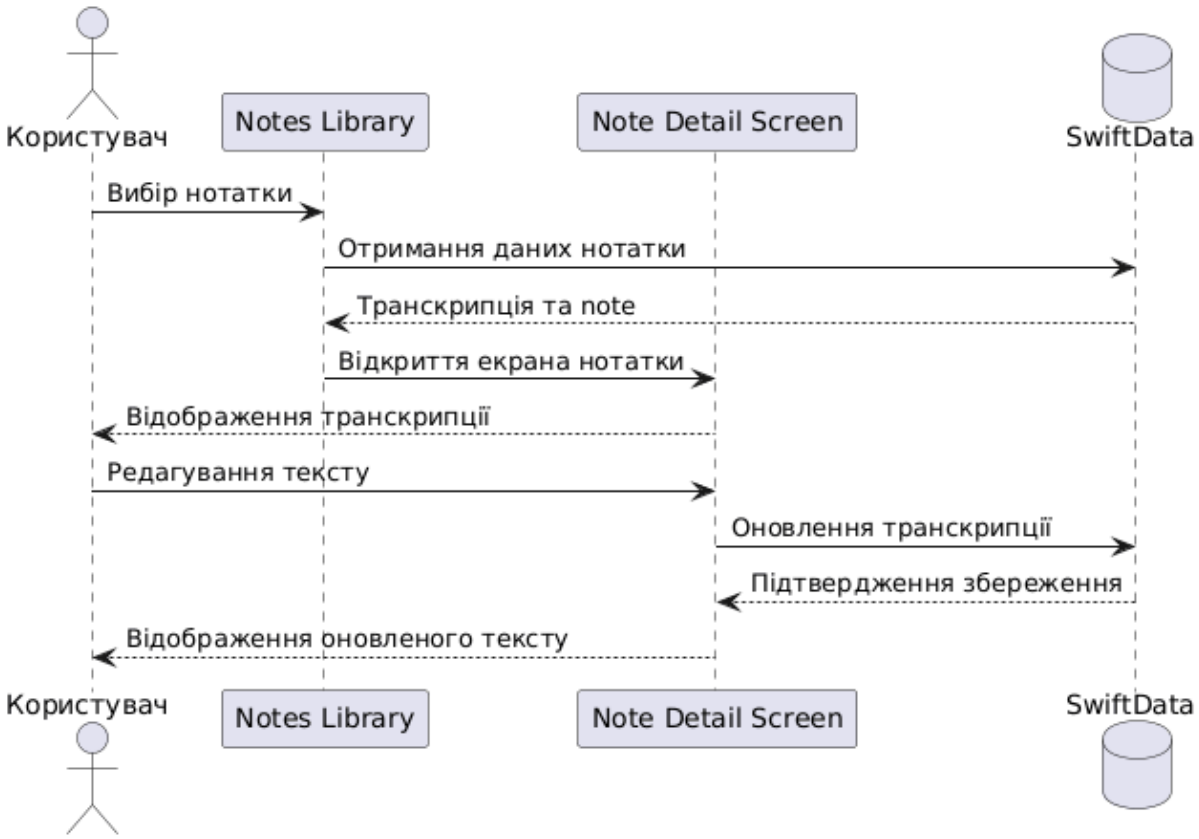


Рисунок 3.3 – Сценарій перегляду та редагування транскрипції

3.3.4 Сценарій AI-перетворення текстових нотаток

Сценарій AI-перетворення текстових нотаток реалізує механізм автоматичної трансформації транскрипцій за допомогою Foundation Models. Після відкриття нотатки користувач отримує можливість обрати необхідний режим обробки тексту, зокрема створення короткого підсумку, структурованого списку тощо. Для виконання AI-обробки застосунок отримує текст транскрипції та передає його разом із параметрами стилю до Foundation Models (рисунок 3.4).

Після завершення генерації система повертає оновлений текстовий результат, який може бути збережений як refined note у локальній базі даних SwiftData. Отриманий результат відображається на екрані нотатки та стає доступним для подальшого перегляду, копіювання або редагування.

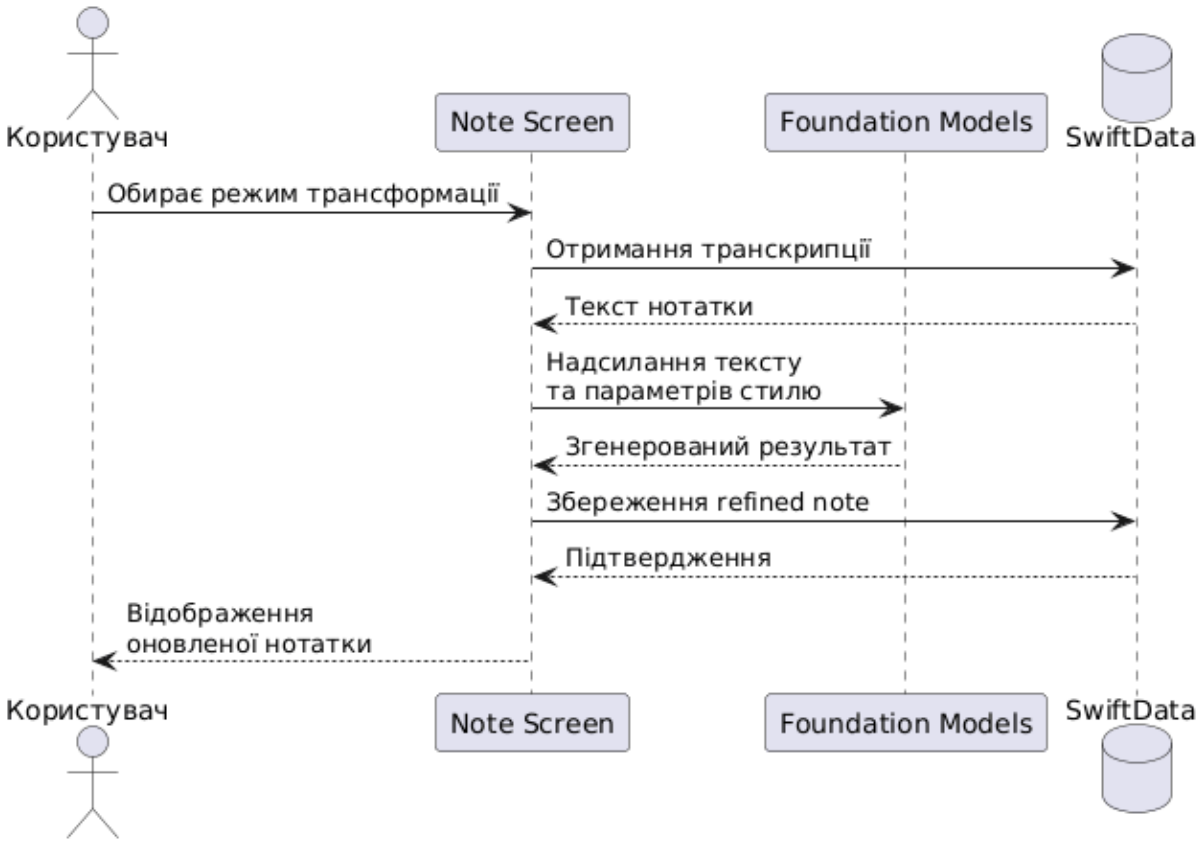


Рисунок 3.4 – Сценарій AI-перетворення текстових нотаток

Такий сценарій суттєво розширює функціональні можливості мобільного застосунку, оскільки дозволяє автоматизувати структурування та покращення текстових нотаток. Це підвищує зручність повторного використання транскрипцій у робочих, навчальних або особистих сценаріях використання.

3.3.5 Схема використання мобільного застосунку

Схема використання мобільного застосунку відображає основні варіанти взаємодії користувача із функціональними можливостями системи. Центральним сценарієм роботи є створення голосової нотатки з подальшим розпізнаванням мовлення у режимі реального часу та збереженням транскрипції. Після формування нотатки користувач отримує доступ до

бібліотеки збережених записів, де може виконувати перегляд, пошук та відкриття окремих нотаток для подальшої роботи з ними (рисунок 3.5). У межах роботи із нотатками застосунок підтримує додаткові сценарії, пов’язані з AI-перетворенням тексту, копіюванням результатів транскрипції та видаленням записів із локального сховища. Такий підхід дозволяє сформувати завершений цикл роботи із голосовими даними – від запису мовлення до подальшого редагування та використання сформованих текстових нотаток.

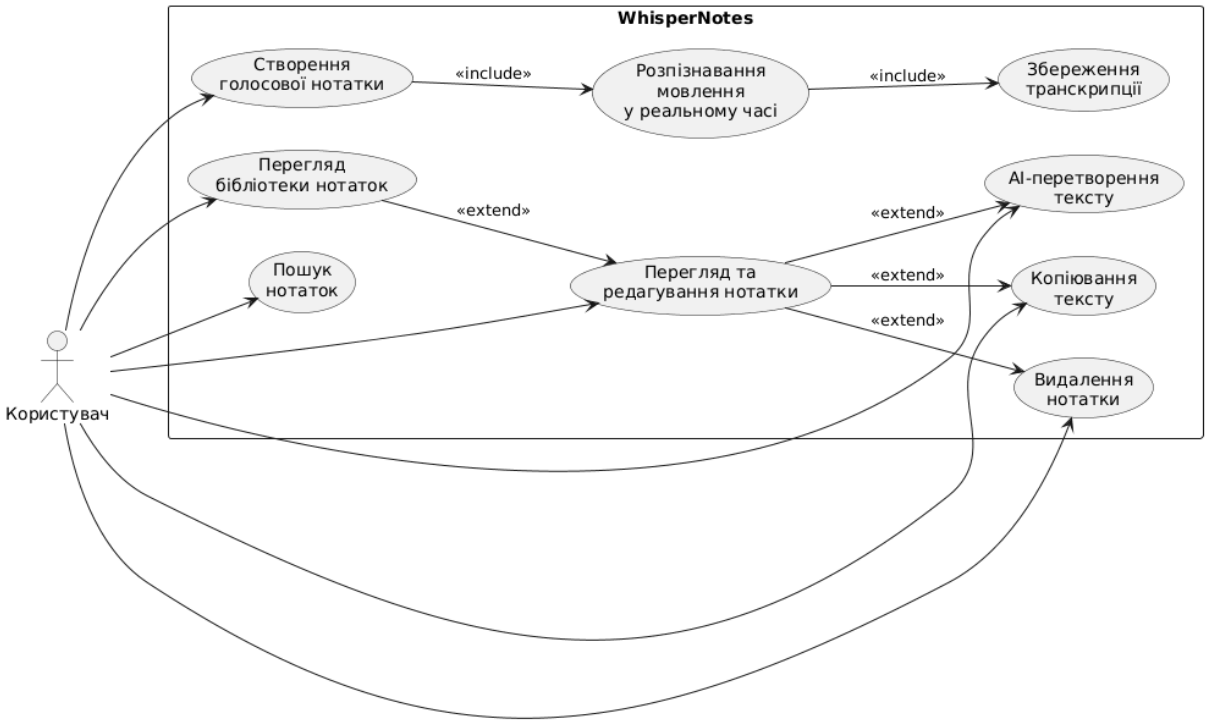


Рисунок 3.5 – Схема використання мобільного застосунку

Представлена схема демонструє взаємозв’язок між основними функціональними можливостями системи та дозволяє наочно відобразити структуру користувацьких сценаріїв у межах мобільного застосунку.

3.4 Проєктування архітектури мобільного застосунку

Проєктування архітектури мобільного застосунку є одним із ключових етапів розробки системи, оскільки саме архітектурна організація визначає взаємодію між окремими програмними компонентами, ефективність обробки даних та можливість подальшого масштабування функціональності. Для застосунків, що використовують моделі машинного навчання та виконують обробку аудіосигналів у режимі реального часу, особливого значення набуває чітке розділення відповідальності між шарами системи та мінімізація залежностей між окремими модулями.

У межах розроблюваного застосунку архітектура (рисунок 3.6) була побудована за багатошаровим принципом із розділенням Presentation Layer, Application Layer, ML Layer та Data Layer. Такий підхід дозволяє ізолювати логіку інтерфейсу користувача від механізмів обробки аудіо, AI-функціональності та локального зберігання даних.

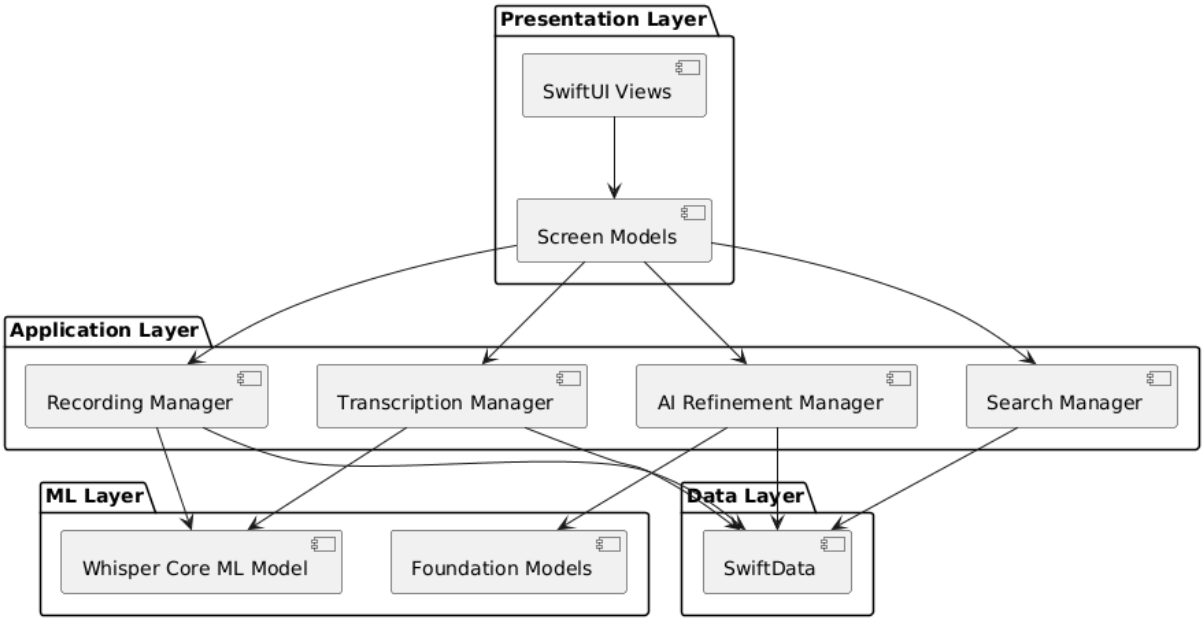


Рисунок 3.6 – Архітектура мобільного застосунку

Presentation Layer відповідає за відображення інтерфейсу користувача та взаємодію між екранами застосунку. У цьому шарі використовуються SwiftUI Views та Screen Models, які реалізують механізми оновлення стану інтерфейсу, обробки користувацьких дій та передачі команд до бізнес-логіки системи. Така організація дозволяє забезпечити реактивну модель роботи інтерфейсу та спростити підтримку окремих екранів застосунку.

Application Layer містить основні програмні менеджери, що реалізують логіку роботи застосунку. До складу цього шару входять модулі Recording Manager, Transcription Manager, AI Refinement Manager та Search Manager. Вони відповідають за керування процесом запису аудіо, взаємодію з моделями машинного навчання, AI-обробку транскрипцій та пошук нотаток у локальному сховищі.

ML Layer реалізує інтеграцію моделей машинного навчання, необхідних для роботи системи розпізнавання мовлення та AI-трансформації тексту. Whisper Core ML Model використовується для локального розпізнавання мовлення, тоді як Foundation Models забезпечують функціональність AI-перетворення текстових нотаток. Використання окремого ML-шару дозволяє ізолювати моделі від бізнес-логіки застосунку та спростити їх подальшу модифікацію або оновлення.

Data Layer відповідає за локальне зберігання нотаток, транскрипцій та AI-оброблених результатів із використанням SwiftData. Цей шар забезпечує швидкий доступ до даних, автономність роботи системи та підтримку механізмів пошуку й оновлення інформації. Використання локального сховища також дозволяє забезпечити конфіденційність користувацьких даних та реалізувати повністю on-device pipeline обробки інформації.

3.4.1 Обґрунтування вибору технологічного стеку

Для реалізації мобільного застосунку було обрано технологічний стек, орієнтований на сучасну екосистему iOS та підтримку локального

виконання моделей машинного навчання. Як основний фреймворк інтерфейсу використовується SwiftUI, що забезпечує реактивну модель побудови UI та спрощує підтримку динамічного оновлення live-транскрипції. Для локального зберігання даних використовується SwiftData, а реалізація системи розпізнавання мовлення базується на інтеграції Whisper-моделі через Core ML та використанні Foundation Models для AI-обробки текстових нотаток.

3.4.2 Архітектурний підхід та організація компонентів системи

Архітектура мобільного застосунку побудована за багатошаровим принципом із розділенням інтерфейсного, прикладного, ML та data-рівнів. Такий підхід дозволяє ізолювати окремі компоненти системи, спростити підтримку програмного коду та забезпечити незалежність між модулями, що відповідають за інтерфейс, бізнес-логіку, AI-обробку та локальне зберігання даних. Взаємодія між компонентами реалізується через окремі менеджери та screen models, які координують процеси запису, транскрипції та AI-перетворення тексту.

3.4.3 Організація on-device ML pipeline для реального часу

On-device ML pipeline організований як послідовний процес локальної обробки аудіосигналу без використання зовнішніх серверів. Після захоплення аудіо система виконує формування аудіофрагментів, їх передачу до Whisper-моделі та отримання часткової транскрипції у режимі реального часу. Отримані результати динамічно відображаються в інтерфейсі застосунку та можуть бути додатково передані до Foundation Models для AI-обробки тексту.

3.4.4 Інтеграція Whisper-моделі через Core ML

Інтеграція Whisper-моделі реалізована через Core ML, що дозволяє використовувати апаратне прискорення iOS-пристрою для виконання локального інференсу нейронної мережі. Конвертація моделі у формат Core ML забезпечує сумісність із Neural Engine та оптимізує швидкодію системи під час обробки аудіоданих у реальному часі. Такий підхід дозволяє мінімізувати затримки транскрипції та забезпечити стабільну роботу застосунку без залежності від мережевих сервісів.

3.4.5 Використання Apple Foundation Models для обробки тексту

Для реалізації AI-перетворення текстових нотаток у застосунку використовуються Apple Foundation Models. Зазначені моделі забезпечують локальну генерацію тексту та дозволяють виконувати структуризацію, скорочення або зміну стилю транскрипцій без передачі даних у хмарне середовище. Використання Foundation Models дозволяє інтегрувати AI-функціональність безпосередньо у мобільний застосунок та підтримувати єдиний on-device pipeline обробки користувацьких даних.

3.5 Проєктування інтерфейсу мобільного застосунку

На етапі проєктування мобільного застосунку було сформовано концепт користувацького інтерфейсу, орієнтований на швидку та інтуїтивно зрозумілу взаємодію із системою розпізнавання мовлення. Основною метою під час створення концепту було забезпечення мінімальної кількості дій для запуску запису голосу, перегляду транскрипції та подальшої роботи із текстовими нотатками. При розробці структури інтерфейсу враховувалися сучасні рекомендації Apple Human Interface Guidelines, а також особливості взаємодії користувача із системами live-транскрипції в реальному часі.

Концепт інтерфейсу побудований на dark-first стилі оформлення з використанням мінімалістичного дизайну, м’яких градієнтів та акцентних елементів керування записом. Центральне місце в структурі застосунку займає екран створення голосової нотатки, який забезпечує швидкий доступ до функцій запису, pause/resume режиму та live-відображення текстової транскрипції. Додатково реалізовано окремий екран бібліотеки нотаток із картковою структурою відображення, що дозволяє зручно переглядати збережені записи та здійснювати навігацію між ними (рисунок 3.7).



Рисунок 3.7 – Концепт дизайну мобільного застосунку

У межах концепту також передбачено підтримку AI-обробки текстових нотаток, що інтегрується безпосередньо у сценарії перегляду транскрипції.

Такий підхід дозволяє забезпечити цілісний користувацький досвід, у якому процес запису мовлення, редагування тексту та використання штучного інтелекту для перетворення нотаток об’єднані в межах єдиної системи взаємодії.

3.5.1 Концепція UX/UI для системи голосових нотаток

Концепція UX/UI мобільного застосунку орієнтована на забезпечення максимально швидкого та природного процесу створення голосових нотаток. Основна увага під час проєктування інтерфейсу приділялася мінімізації кількості дій користувача, спрощенню навігації між екранами та збереженню фокусу на ключових функціях системи – записі мовлення та перегляді транскрипції. Для формування сучасного користувацького досвіду використовувалися принципи мінімалістичного дизайну, плавні анімації та нативні компоненти екосистеми iOS.

3.5.2 Проєктування екрана запису та live-транскрипції

Екран запису голосу реалізований як центральний елемент мобільного застосунку та забезпечує швидкий доступ до функцій створення нотатки. У межах екрана передбачено відображення live-транскрипції у режимі реального часу, елементів керування записом, таймера та візуалізації аудіосигналу. Така структура дозволяє користувачу одночасно контролювати процес запису та переглядати результати розпізнавання.

3.5.3 Проєктування бібліотеки нотаток та екрана перегляду

Бібліотека нотаток реалізована у вигляді карткового відображення збережених транскрипцій, що забезпечує швидкий доступ до окремих записів та покращує навігацію всередині застосунку. Для кожної нотатки відображається короткий фрагмент тексту, дата створення та додаткові елементи взаємодії. Екран детального перегляду нотатки забезпечує можливість перегляду транскрипції, редагування тексту, копіювання результатів та запуску AI-перетворення текстових даних.

3.5.4 Візуалізація станів запису та обробки аудіо

Для забезпечення зрозумілої взаємодії користувача із системою у мобільному застосунку реалізовано механізми візуалізації поточного стану запису та обробки аудіосигналу. Відображення активного стану запису дозволяє користувачу контролювати процес захоплення мовлення та своєчасно реагувати на зміни режимів роботи системи. Для цього в інтерфейсі використовуються анімовані елементи, індикатори активності та зміна візуального стану кнопок керування записом.

Під час роботи системи в режимі live-транскрипції користувачу відображається waveform-візуалізація аудіосигналу, яка динамічно оновлюється відповідно до вхідного мовлення. Додатково реалізовано індикатори станів pause, active recording та processing, що дозволяють наочно демонструвати поточний етап роботи системи. Такий підхід покращує сприйняття процесу розпізнавання мовлення та формує більш природний користувацький досвід взаємодії із застосунком.

Окрему увагу приділено візуалізації процесів завершення запису та AI-обробки тексту. Після завершення захоплення аудіо інтерфейс переходить у стан фінальної обробки транскрипції, а під час виконання AI-трансформації відображаються індикатори генерації тексту. Використання плавних анімацій та зміни станів інтерфейсу дозволяє зробити процес роботи системи більш передбачуваним для користувача та зменшити когнітивне навантаження під час взаємодії із застосунком.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Вибір технологій та засобів реалізації

Під час реалізації мобільного застосунку важливим етапом є вибір технологій та програмних засобів, що забезпечують підтримку функціональних вимог системи, стабільність роботи та можливість інтеграції моделей машинного навчання. Оскільки розроблюваний застосунок орієнтований на виконання live-транскрипції та AI-обробки тексту без використання хмарної інфраструктури, особливого значення набуває підтримка on-device обчислень та ефективне використання апаратних ресурсів мобільного пристрою.

Для реалізації системи було використано сучасний стек технологій екосистеми Apple, що включає Swift, SwiftUI, Core ML, SwiftData та Foundation Models. Обрані засоби розробки забезпечують інтеграцію моделей машинного навчання безпосередньо у мобільний застосунок, підтримку локального зберігання даних та створення сучасного користувацького інтерфейсу відповідно до рекомендацій Apple Human Interface Guidelines.

4.1.1 Використання Swift та SwiftUI для створення iOS-застосунку

Мовою програмування для реалізації мобільного застосунку було обрано Swift, яка є основною мовою розробки програмного забезпечення для платформ Apple. Використання Swift забезпечує високу продуктивність, безпечну роботу з пам’яттю та підтримку сучасних механізмів асинхронного програмування. Додатковою перевагою є глибока інтеграція мови з екосистемою iOS та підтримка фреймворків машинного навчання Apple.

Для побудови користувацького інтерфейсу використовується SwiftUI, що дозволяє реалізувати реактивну модель оновлення UI та спростити організацію взаємодії між екранами застосунку. SwiftUI забезпечує підтримку анімацій, динамічного оновлення live-транскрипції та адаптацію інтерфейсу під різні типи iOS-пристроїв. Використання декларативного підходу також позитивно впливає на підтримуваність та масштабованість програмного коду.

4.1.2 Застосування Core ML для on-device розпізнавання мовлення

Для реалізації системи локального розпізнавання мовлення використовується Core ML – фреймворк машинного навчання екосистеми Apple. Core ML забезпечує можливість інтеграції нейронних моделей безпосередньо у мобільний застосунок та підтримує виконання inference-процесів без використання зовнішніх серверів. Це дозволяє реалізувати повністю автономний pipeline розпізнавання мовлення з високим рівнем конфіденційності користувацьких даних.

Важливою перевагою використання Core ML є підтримка апаратного прискорення через Neural Engine, GPU та CPU iOS-пристрою. Завдяки цьому система здатна виконувати live-транскрипцію у режимі реального часу з мінімальними затримками обробки. Використання Core ML також спрощує процес оптимізації моделей для мобільного середовища та забезпечує стабільну роботу застосунку на різних поколіннях iPhone.

4.1.3 Інтеграція Whisper-моделі у середовище iOS

У межах розроблюваного застосунку для розпізнавання мовлення використовується Whisper-модель, адаптована для роботи у середовищі iOS через Core ML. Інтеграція моделі передбачає конвертацію нейронної мережі у формат Core ML та організацію локального inference-процесу під час

обробки аудіоданих. Такий підхід дозволяє виконувати розпізнавання мовлення без використання зовнішніх API або хмарної інфраструктури.

Whisper-модель забезпечує підтримку високої якості транскрипції навіть при роботі з довільним мовленням та різними умовами запису аудіо. Додатково інтеграція моделі у середовище iOS дозволяє використовувати апаратне прискорення системи та підтримувати режим live-транскрипції в реальному часі. Це створює основу для реалізації повністю локальної системи голосових нотаток.

4.1.4 Використання SwiftData для локального збереження транскрипцій

Для організації локального зберігання транскрипцій та текстових нотаток використовується SwiftData. Обраний фреймворк забезпечує сучасний механізм роботи з локальними моделями даних у середовищі Swift та спрощує реалізацію CRUD-операцій у мобільному застосунку. Використання SwiftData дозволяє ефективно організувати структуру збереження нотаток, AI-оброблених результатів та метаданих запису.

Додатковою перевагою SwiftData є інтеграція із SwiftUI та підтримка реактивного оновлення інтерфейсу при зміні даних. Це дозволяє автоматично оновлювати бібліотеку нотаток, результати пошуку та екрани перегляду транскрипцій без необхідності ручного оновлення стану UI. Локальне зберігання даних також забезпечує автономність роботи застосунку та підвищує рівень конфіденційності інформації користувача.

4.1.5 Використання Apple Foundation Models для інтелектуальної обробки нотаток

Для реалізації AI-перетворення текстових нотаток у мобільному застосунку використовуються Apple Foundation Models. Застосування

Foundation Models дозволяє виконувати локальну генерацію тексту, структуризацію транскрипцій та зміну стилю нотаток без використання сторонніх AI-сервісів. Це забезпечує інтеграцію AI-функціональності безпосередньо у середовище iOS та підтримку повністю on-device обробки даних.

За допомогою Foundation Models користувач отримує можливість автоматичного формування підсумків, списків, електронних листів та інших форматів текстового представлення інформації. Використання локальної AI-обробки також дозволяє зменшити затримки генерації результатів та забезпечити високий рівень конфіденційності при роботі із транскрипціями голосових нотаток.

4.2 Реалізація основної функціональності застосунку

Реалізація основної функціональності мобільного застосунку охоплює побудову програмних компонентів, необхідних для запису аудіо, розпізнавання мовлення, формування текстових нотаток та AI-обробки транскрипцій. У межах реалізації особлива увага приділялася забезпеченню роботи системи в режимі реального часу, мінімізації затримок обробки та підтримці повністю локального виконання всіх основних процесів без використання хмарної інфраструктури. Такий підхід дозволяє забезпечити автономність роботи застосунку та підвищити рівень конфіденційності користувацьких даних.

Функціональність системи реалізована у вигляді окремих програмних модулів, кожен із яких відповідає за визначений етап обробки інформації. До основних компонентів належать модуль захоплення аудіосигналу, система live-транскрипції, механізм локального збереження нотаток та AI-модуль трансформації тексту. Взаємодія між зазначеними компонентами організована таким чином, щоб забезпечити безперервний процес роботи із

голосовими нотатками – від моменту початку запису до перегляду та редагування сформованої транскрипції.

Під час реалізації застосунку також було враховано особливості роботи мобільних пристроїв iOS, зокрема обмеження використання пам’яті, необхідність підтримки плавного інтерфейсу та ефективного використання апаратного прискорення системи. Використання SwiftUI, Core ML, SwiftData та Foundation Models дозволило сформувати єдину інтегровану архітектуру, у межах якої всі основні функції системи працюють у межах локального on-device pipeline обробки даних.

4.2.1 Реалізація запису та обробки мовлення

Реалізація запису та обробки мовлення є початковим етапом роботи мобільного застосунку, оскільки саме цей компонент забезпечує отримання аудіосигналу з мікрофона iOS-пристрою. У межах реалізації було передбачено створення окремого менеджера запису, який відповідає за налаштування аудіосесії, запуск аудіодвигуна, отримання аудіобуферів та контроль поточного стану запису. Такий підхід дозволяє відокремити логіку роботи з аудіо від інтерфейсної частини застосунку та спростити подальшу передачу аудіоданих до модуля транскрибування.

Основна логіка запису мовлення реалізується через AVAudioEngine, який забезпечує доступ до вхідного аудіопотоку та дозволяє встановити tap на inputNode для отримання аудіобуферів у реальному часі. Під час активного запису кожен отриманий AVAudioPCMBuffer передається до callback-функції onAudioBufferReceived, після чого може бути використаний модулем розпізнавання мовлення. Реалізацію менеджера запису аудіосигналу наведено у лістингу 4.1.

У програмному коді також передбачено керування станами запису, зокрема запуск, паузу, відновлення та завершення запису. Це дозволяє користувачу гнучко керувати процесом створення голосової нотатки без

втрати поточного стану системи. Додатково в менеджері реалізовано облік тривалості запису, що надалі використовується під час формування метаданих текстової нотатки. Така організація забезпечує стабільну роботу аудіопотоку та створює основу для подальшої live-транскрипції мовлення.

Лістинг 4.1 – Програмний код менеджера запису аудіосигналу

```
@Observable
final class RecordingManager {
    private let audioEngine = AVAudioEngine()
    private let inputBus: AVAudioNodeBus = 0
    private var inputFormat: AVAudioFormat?
    private(set) var isRecording = false
    private(set) var isPaused = false
    private(set) var recordingDuration: TimeInterval = 0
    private var startedAt: Date?
    private var timer: Timer?
    var onAudioBufferReceived: ((AVAudioPCMBuffer) -> Void)?
    func startRecording() throws {
        let session = AVAudioSession.sharedInstance()
        try session.setCategory(.record, mode: .measurement, options: [.duckOthers])
        try session.setActive(true, options: .notifyOthersOnDeactivation)
        let inputNode = audioEngine.inputNode
        let format = inputNode.outputFormat(forBus: inputBus)
        inputFormat = format
        inputNode.removeTap(onBus: inputBus)
        inputNode.installTap(onBus: inputBus, bufferSize: 1024, format: format) { [weak self] buffer, _ in
            guard let self, self.isRecording, !self.isPaused else { return }
            self.onAudioBufferReceived?(buffer)
```

Продовження лістингу 4.1

```
            }
            audioEngine.prepare()
            try audioEngine.start()
            isRecording = true
            isPaused = false
            startedAt = Date()
            startTimer()
        }
        private func startTimer() {
            timer?.invalidate()
            timer = Timer.scheduledTimer(withTimeInterval: 1,
repeats: true) { [weak self] _ in
                guard let self, let startedAt = self.startedAt,
!self.isPaused else { return }
                self.recordingDuration                    +=
Date().timeIntervalSince(startedAt)
                self.startedAt = Date()
            }
        }
        private func stopTimer() {
            timer?.invalidate()
            timer = nil
        }
    }
```

4.2.2 Реалізація транскрибування мовлення у реальному часі

Реалізація транскрибування мовлення у реальному часі ґрунтується на поетапній обробці аудіофрагментів, які надходять від модуля запису. Після отримання аудіобуфера система не очікує завершення повного запису, а накопичує фрагменти мовлення та передає їх до сервісу розпізнавання. Такий підхід дозволяє поступово формувати текстову транскрипцію під час

мовлення користувача та забезпечувати ефект live-відображення результату в інтерфейсі застосунку.

Основна логіка транскрибування реалізована в окремому менеджері LiveTranscriptionManager, який відповідає за приймання аудіобуферів, формування набору аудіофрагментів, запуск асинхронного процесу розпізнавання та оновлення часткової транскрипції. У програмному коді передбачено перевірку стану isTranscribing, що запобігає одночасному запуску кількох процесів обробки та зменшує ризик перевантаження системи. Реалізацію обробки аудіофрагментів та live-транскрипції наведено у лістингу 4.2.

Лістинг 4.2 – Програмний код, що реалізує обробки аудіофрагментів та live-транскрипції

```
@Observable
final class LiveTranscriptionManager {
    private let whisperService: WhisperCoreMLService
    private var audioChunks: [AVAudioPCMBuffer] = []
    private var transcriptionTask: Task<Void, Never>?
    private(set) var partialTranscript = ""
    private(set) var finalTranscript = ""
    private(set) var isTranscribing = false
    init(whisperService: WhisperCoreMLService) {
        self.whisperService = whisperService
    }
    func appendAudioBuffer(_ buffer: AVAudioPCMBuffer) {
        audioChunks.append(buffer)
        if audioChunks.count >= 8 {
            let chunks = audioChunks
            audioChunks.removeAll()
            transcribe(chunks)
        }
    }
```

Продовження лістингу 4.2

```
        private func transcribe(_ buffers: [AVAudioPCMBuffer])
{
            guard !isTranscribing else { return }
            isTranscribing = true
            transcriptionTask = Task { [weak self] in
                guard let self else { return }
                do {
                    let text = try await
whisperService.transcribe(buffers: buffers)
                    await MainActor.run {
                        if !text.isEmpty {
                            self.partialTranscript += text + "
"
                            self.finalTranscript =
self.partialTranscript.trimmingCharacters(in:
.whitespacesAndNewlines)
                        }
                        self.isTranscribing = false
                    }
                } catch {
                    await MainActor.run {
                        self.isTranscribing = false
                    }
                }
            }
        }
```

Після завершення обробки чергового аудіофрагмента розпізнаний текст додається до partialTranscript, а фінальна версія транскрипції оновлюється у finalTranscript. Окремо реалізовано метод finishTranscription, який виконує обробку залишкових аудіоданих після завершення запису.

4.2.3 Реалізація збереження та перегляду нотаток

Реалізація збереження та перегляду нотаток забезпечує завершення основного сценарію роботи із голосовою транскрипцією. Після завершення запису та формування фінального тексту система створює окремий об’єкт нотатки, у якому зберігаються назва, транскрипція, AI-оброблений текст, тривалість запису, дата створення та дата останнього оновлення. Така структура дозволяє не лише зберігати результат розпізнавання мовлення, а й підтримувати подальше редагування, пошук і повторне використання текстових даних.

Для локального збереження інформації використовується SwiftData, що дозволяє організувати роботу з моделями даних без необхідності використання зовнішньої бази або хмарного сховища. У межах менеджера нотаток реалізовано основні операції над записами: створення нової нотатки, оновлення транскрипції, збереження AI-обробленого тексту та видалення непотрібного запису. Програмну реалізацію моделі голосової нотатки та механізму її збереження наведено у лістингу 4.3.

Лістинг 4.3 – Програмний код моделі та збереження текстової нотатки

```
@Model
final class VoiceNote {
    var id: UUID
    var title: String
    var transcript: String
    var refinedText: String
    var duration: TimeInterval
    var createdAt: Date
    var updatedAt: Date
    init(title: String, transcript: String, refinedText:
String = "", duration: TimeInterval) {
        self.id = UUID()
        self.title = title
```

Продовження лістингу 4.3

```
            self.transcript = transcript
            self.refinedText = refinedText
            self.duration = duration
            self.createdAt = Date()
            self.updatedAt = Date()
        }
    }
    @Observable
    final class NotesManager {
        private let modelContext: ModelContext
        init(modelContext: ModelContext) {
            self.modelContext = modelContext
        }
        func saveNote(transcript: String, duration:
TimeInterval) {
            let preparedText =
transcript.trimmingCharacters(in: .whitespacesAndNewlines)
            guard !preparedText.isEmpty else { return }
            let title = makeTitle(from: preparedText)
            let note = VoiceNote(
                title: title,
                transcript: preparedText,
                duration: duration
            )
            modelContext.insert(note)
            try? modelContext.save()
        }
    }
```

Окремим елементом реалізації є перегляд бібліотеки нотаток, у якій користувач може працювати зі збереженими транскрипціями. Для цього застосовується список або сітка карток, що відображають коротку інформацію про кожну нотатку та дозволяють перейти до екрана детального

перегляду. Також передбачено пошук за назвою, текстом транскрипції та AI-обробленим результатом, що підвищує зручність роботи з великою кількістю збережених голосових записів.

4.2.4 Реалізація інтелектуальної обробки тексту

Реалізація інтелектуальної обробки тексту забезпечує розширення базового сценарію транскрибування мовлення та дозволяє перетворювати отриману транскрипцію у більш структурований і зручний для подальшого використання формат.

Після збереження або відкриття голосової нотатки користувач може обрати один із режимів обробки, наприклад формування короткого підсумку, списку, електронного листа або нотаток зустрічі. Такий підхід дає змогу використовувати розпізнаний текст не лише як сирий результат транскрипції, а як готовий інформаційний матеріал.

Основна логіка AI-обробки реалізована в окремому менеджері NoteRefinementManager, який отримує текст транскрипції, формує prompt із урахуванням обраного режиму та стилю, після чого передає його до Apple Foundation Models. У результаті модель генерує оновлений текст, який зберігається у полі refinedText відповідної нотатки. Програмну реалізацію сервісу AI-перетворення транскрипції наведено у лістингу 4.4.

У програмному коді також передбачено обробку стану генерації та можливих помилок, що дозволяє коректно відображати процес AI-перетворення в інтерфейсі застосунку.

Після успішної генерації результат зберігається у локальному сховищі SwiftData, а дата оновлення нотатки змінюється відповідно до моменту виконання операції. Завдяки цьому користувач може повторно відкривати AI-оброблені версії транскрипцій без необхідності повторної генерації тексту.

Лістинг 4.4 – Програмний код сервісу AI-перетворення транскрипції

```
@Observable
final class NoteRefinementManager {
    private let modelContext: ModelContext
    private(set) var isGenerating = false
    private(set) var errorMessage: String?
    init(modelContext: ModelContext) {
        self.modelContext = modelContext
    }
    func refine(note: VoiceNote, mode: RefinementMode,
tone: RefinementTone) async {
        let sourceText =
note.transcript.trimmingCharacters(in: .whitespacesAndNewlines)
        guard !sourceText.isEmpty else { return }
        await MainActor.run {
            isGenerating = true
            errorMessage = nil
        }
        do {
            let prompt = buildPrompt(text: sourceText,
mode: mode, tone: tone)
            let session = LanguageModelSession()
            let response = try await session.respond(to:
prompt)
            await MainActor.run {
                note.refinedText =
response.content.trimmingCharacters(in:
.whitespacesAndNewlines)
                note.updatedAt = Date()
                try? modelContext.save()
                isGenerating = false
            }
        } catch {
            await MainActor.run {
                errorMessage = "Unable to refine note"
```

Продовження лістингу 4.4

```
                    isGenerating = false
                }
            }
        }
        private func buildPrompt(text: String, mode: RefinementMode, tone: RefinementTone) -> String {
            """
            Transform the following transcript into \(mode.promptName).
            Use a \(tone.promptName) tone.
            Keep the original meaning and remove speech filler words.
            Transcript:
            \(text)
            """
        }
    }
```

4.3 Реалізація інтерфейсу користувача

Реалізація інтерфейсу користувача була виконана з використанням SwiftUI, що дозволило побудувати застосунок за декларативним принципом та забезпечити динамічне оновлення станів екранів. Основна структура інтерфейсу передбачає поділ застосунку на два ключові розділи: екран запису голосової нотатки та бібліотеку збережених нотаток. Така організація відповідає основному сценарію використання системи, за якого користувач спочатку створює транскрипцію, а потім переглядає, редагує або повторно використовує збережені текстові записи.

Навігаційна структура застосунку реалізована через TabView, що забезпечує швидке перемикання між основними функціональними екранами. Перший розділ відповідає за створення голосової нотатки, а другий – за роботу з бібліотекою транскрипцій. Використання окремого

enum для визначення вкладок дозволяє зробити код більш структурованим та зручним для подальшого розширення. Програмну реалізацію головної навігації мобільного застосунку наведено у лістингу 4.5.

Лістинг 4.5 – Програмний код головної навігації застосунку

```
struct MainTabView: View {
    @State private var selectedTab: AppTab = .record
    var body: some View {
        TabView(selection: $selectedTab) {
            RecordingView()
                .tabItem {
                    Label("Record",           systemImage:
"mic.fill")
                }
                .tag(AppTab.record)
            NotesLibraryView()
                .tabItem {
                    Label("Notes",            systemImage:
"note.text")
                }
                .tag(AppTab.notes)
        }
        .tint(.orange)
    }
}
enum AppTab {
    case record
    case notes
}
```

Окрему увагу було приділено реалізації екрана запису та перегляду live-транскрипції. У межах цього екрана передбачено відображення поточного стану запису, текстової області для результату розпізнавання мовлення та кнопки керування процесом запису. Залежно від значення

isRecording інтерфейс змінює підпис, іконку та функціональний стан основної кнопки, що дозволяє користувачу інтуїтивно розуміти поточний режим роботи системи. Відповідний фрагмент програмної реалізації наведено у лістингу 4.6.

Лістинг 4.6 – Програмний код екрана запису та перегляду транскрипції

```
struct RecordingView: View {
    @State private var transcript = ""
    @State private var isRecording = false
    var body: some View {
        VStack(spacing: 24) {
            Text(isRecording ? "Listening" : "Record")
                .font(.headline)
                .fill(.thinMaterial)
                .overlay {
                    Text(transcript.isEmpty ? "Capture a thought." : transcript)
                        .padding(24)
.frame(maxWidth: .infinity, alignment: .leading)
                }
                .frame(height: 280)
            Button {
                isRecording.toggle()
            } label: {
                Label(isRecording ? "Stop" : "Start", systemImage: isRecording ? "stop.fill" : "mic.fill")
                    .frame(maxWidth: .infinity)
                    .padding()
.background(.orange, in: Rectangle(cornerRadius: 18))
                    .foregroundStyle(.white)}}
        .padding()}
}
```

Застосований підхід дозволив сформувати простий і зрозумілий інтерфейс, у якому основні дії користувача зосереджені навколо запису мовлення та роботи із текстовими нотатками. Мінімалістична структура екранів зменшує кількість зайвих переходів між розділами та забезпечує швидкий доступ до ключових функцій застосунку. Завдяки використанню SwiftUI інтерфейс залишається гнучким, легко підтримується та може бути розширений додатковими елементами, зокрема індикаторами обробки аудіо, waveform-візуалізацією або кнопками AI-перетворення тексту.

4.3.1 Головний екран застосунку

Головний екран мобільного застосунку є центральною точкою взаємодії користувача із системою розпізнавання мовлення. Його основне призначення полягає у забезпеченні швидкого доступу до створення голосової нотатки без необхідності переходу між додатковими розділами або виконання зайвих дій. Такий підхід відповідає загальній концепції застосунку, у межах якої користувач повинен мати можливість швидко розпочати запис мовлення та надалі отримати текстову транскрипцію.

На головному екрані передбачено таймер запису, кнопку переходу до налаштувань, центральну область для відображення стану системи та основну кнопку запуску запису. У стані очікування користувач бачить повідомлення із пропозицією почати запис, а також позначку приватності, яка підкреслює локальний характер обробки даних (рисунок 4.1).

Окрему роль на головному екрані відіграє нижня навігаційна панель, яка забезпечує швидкий перехід між режимами Record та Notes. Це дозволяє користувачу не лише запускати запис мовлення, а й одразу переходити до бібліотеки збережених транскрипцій. Візуальна ієрархія екрана побудована таким чином, щоб основна кнопка Start Recording була найбільш помітним елементом інтерфейсу.

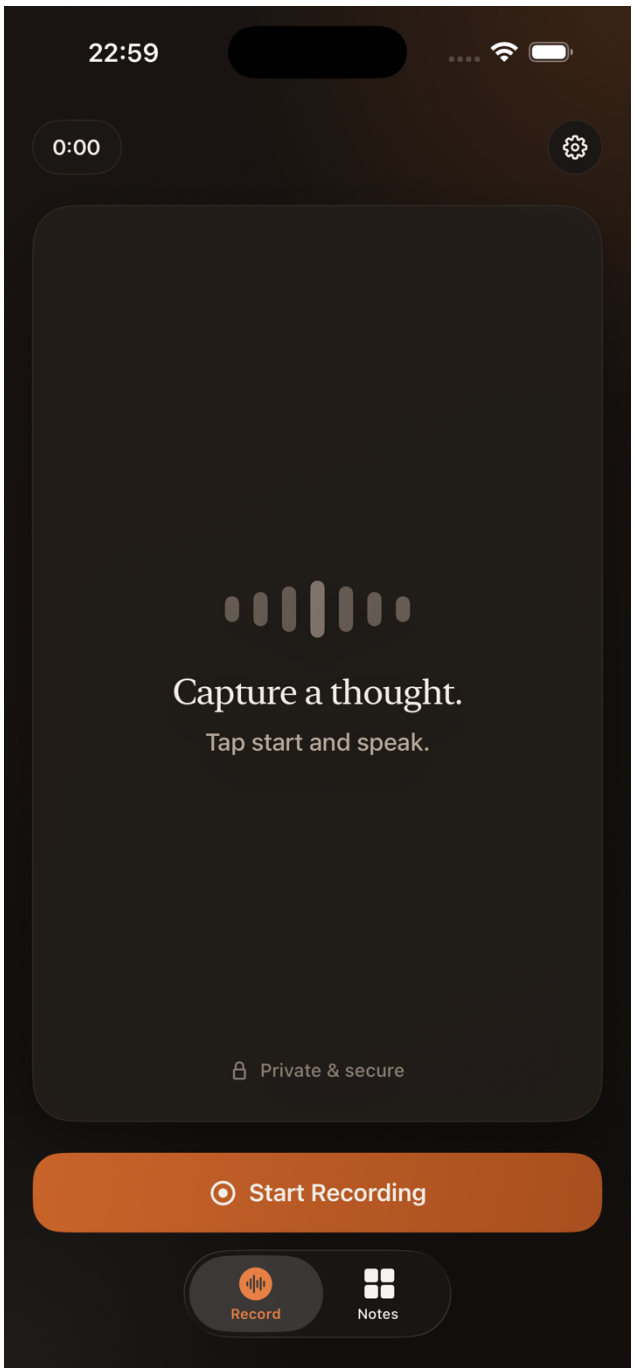


Рисунок 4.1 – Головний екран застосунку

Реалізація головного екрана базується на компонентах SwiftUI, що забезпечують декларативне оновлення інтерфейсу залежно від поточного стану застосунку. Завдяки цьому зміна режимів запису, оновлення таймера та подальше відображення транскрипції можуть виконуватися без ручного оновлення окремих UI-компонентів.

4.3.2 Інтерфейс запису та транскрипції

Інтерфейс запису та транскрипції відображає (рисунок 4.2) активний стан роботи мобільного застосунку, у якому система виконує захоплення аудіосигналу та поступове формування текстового результату.

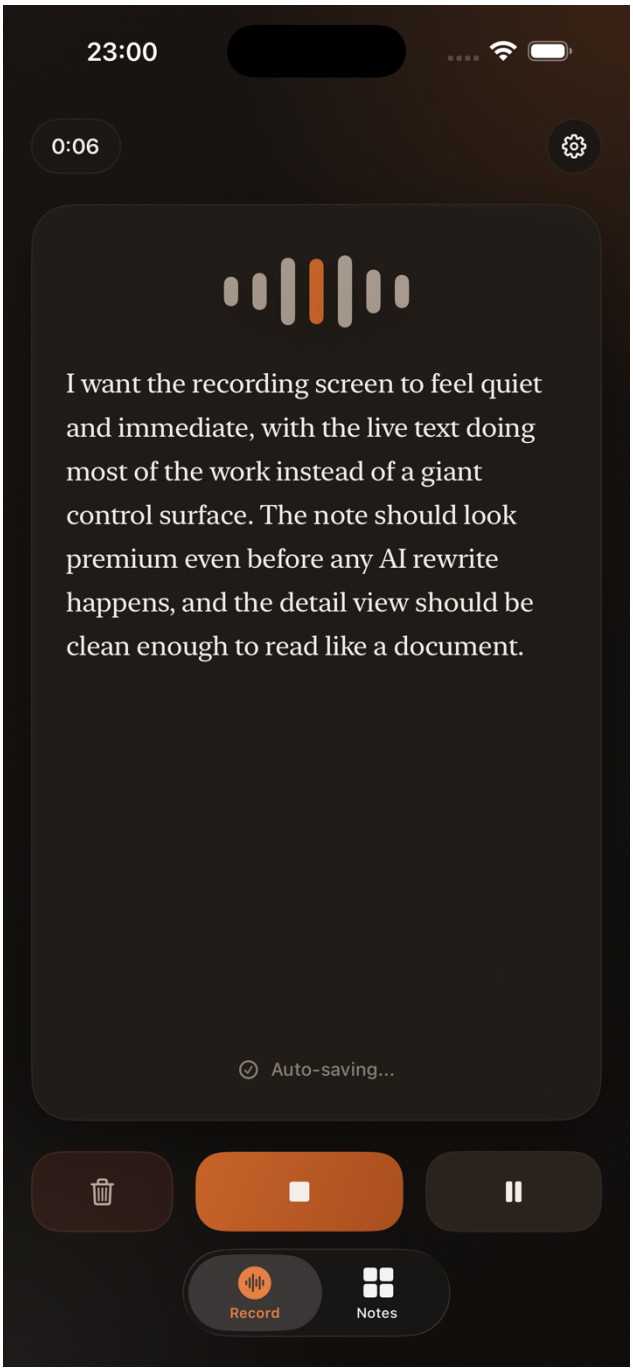


Рисунок 4.2 – Інтерфейс запису та транскрипції

Після запуску запису головний екран змінює свій стан: замість початкового повідомлення користувач бачить розпізнаний текст, таймер запису, waveform-візуалізацію та елементи керування процесом аудіозапису. Такий підхід дозволяє користувачу одразу контролювати як сам факт запису, так і якість live-транскрипції.

Центральним елементом екрана є велика текстова область, у якій у реальному часі відображається розпізнане мовлення (рисунок 4.2). Це дозволяє не чекати завершення запису для перевірки результату, а бачити сформовану транскрипцію безпосередньо під час диктування. Додатково під текстовим блоком відображається статус auto-saving, який інформує користувача про автоматичне збереження поточного стану нотатки.

У нижній частині екрана розміщено основні кнопки керування записом: видалення, завершення та пауза. Кнопка завершення запису візуально виділена акцентним кольором, оскільки вона є головною дією під час активного сценарію транскрибування. Наявність окремої кнопки паузи дозволяє тимчасово зупинити запис без втрати вже сформованої транскрипції, а кнопка видалення забезпечує можливість швидкого скасування поточної нотатки.

Візуальна організація екрана побудована так, щоб основна увага користувача була зосереджена на тексті транскрипції, а не на другорядних елементах керування. Темна кольорова схема, м’які контейнери та обмежена кількість кнопок дозволяють зменшити когнітивне навантаження під час запису мовлення. Це особливо важливо для застосунку, призначеного для швидкого збереження думок, ідей або робочих нотаток у режимі реального часу.

4.3.3 Інтерфейс перегляду та пошуку нотаток

Інтерфейс перегляду та пошуку нотаток (рисунок 4.3) призначений для роботи зі збереженими транскрипціями після завершення запису

мовлення. У цьому розділі користувач отримує доступ до бібліотеки голосових нотаток, які відображаються у вигляді карток із коротким фрагментом тексту, назвою, датою створення та тривалістю запису. Такий формат подання дозволяє швидко переглядати зміст нотаток без необхідності відкривати кожен запис окремо.

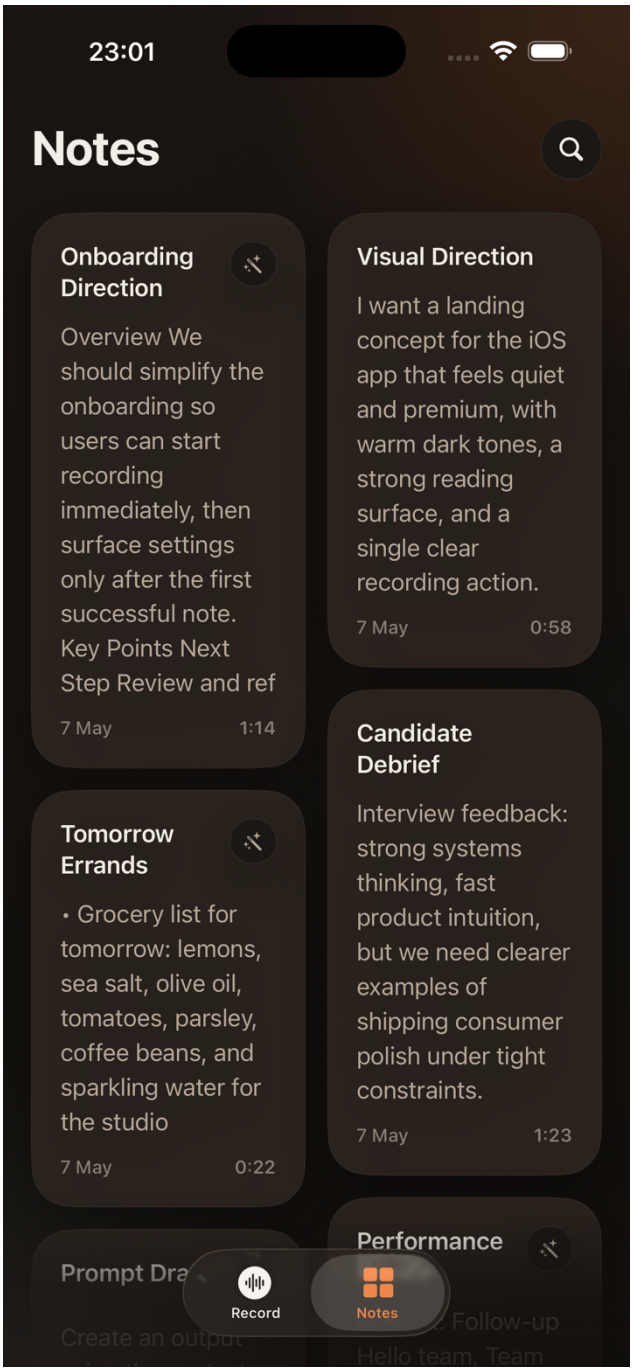


Рисунок 4.3 – Інтерфейс перегляду та пошуку нотаток

Для покращення навігації між збереженими матеріалами було використано карткову структуру з нерівномірним розміщенням елементів, що створює візуально зручну бібліотеку текстових записів (рисунок 4.3). Кожна картка містить достатньо інформації для попереднього розуміння змісту нотатки, а натискання на неї дозволяє перейти до детального перегляду транскрипції або AI-обробленого тексту. Це забезпечує швидкий доступ до потрібного матеріалу та спрощує роботу з великою кількістю записів.

У верхній частині екрана розміщено заголовок розділу Notes та кнопку пошуку. Пошук дозволяє знаходити нотатки за назвою, текстом транскрипції або збереженим AI-результатом, що є важливим для застосунку, у якому кількість голосових записів може поступово збільшуватися. Завдяки цьому користувач може швидко повернутися до потрібної інформації без ручного перегляду всієї бібліотеки.

Нижня навігаційна панель зберігає швидкий перехід між екранами Record та Notes, що підтримує основний сценарій використання застосунку. Користувач може швидко перейти від перегляду збережених нотаток до створення нового запису, не втрачаючи контекст роботи. Така організація інтерфейсу робить застосунок зручним для регулярного використання та відповідає логіці швидкого створення, збереження і повторного використання голосових нотаток.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто поставленої мети – розроблено iOS-застосунок для on-device розпізнавання мовлення в реальному часі з використанням Core ML. У межах роботи було виконано аналіз предметної галузі, досліджено сучасні підходи до автоматичного розпізнавання мовлення, розглянуто особливості використання Whisper-моделей, Core ML, SwiftUI, SwiftData та Apple Foundation Models у мобільному середовищі. На основі проведеного аналізу було сформовано вимоги до системи, спроєктовано функціональну модель, сценарії взаємодії користувача із застосунком, архітектуру програмного рішення та концепт користувацького інтерфейсу.

Поставлені завдання кваліфікаційної роботи виконано у повному обсязі. Було реалізовано основну функціональність мобільного застосунку, зокрема запис мовлення з мікрофона iOS-пристрою, обробку аудіосигналу, транскрибування мовлення у режимі реального часу, локальне збереження текстових нотаток, перегляд бібліотеки транскрипцій, пошук записів та інтелектуальну обробку тексту. Реалізований застосунок забезпечує повний цикл роботи з голосовими нотатками – від початку запису до формування, збереження та подальшого використання текстового результату.

Досягнуті кількісні показники роботи визначаються реалізацією основних функціональних модулів системи. У застосунку передбачено два основні розділи інтерфейсу – Record та Notes, реалізовано чотири ключові сценарії взаємодії користувача із системою: створення голосової нотатки, live-транскрибування мовлення, перегляд і пошук нотаток, AI-перетворення тексту. Також було сформовано багатошарову архітектуру застосунку, що включає presentation layer, application layer, ML layer та data layer. Така структура забезпечує логічне розділення відповідальності між компонентами та створює основу для подальшого розвитку системи.

Практична цінність розробленого застосунку полягає у можливості швидкого перетворення усного мовлення користувача на текстові нотатки без передачі аудіоданих у зовнішнє середовище. Система може бути використана для фіксації ідей, робочих нотаток, навчальних матеріалів, коротких диктувань, підготовки текстових підсумків та подальшої структуризації отриманих транскрипцій. Додаткова інтеграція Apple Foundation Models дозволяє перетворювати сирий текст розпізнавання у більш зручні формати, зокрема підсумки, списки, електронні листи або структуровані нотатки.

Рекомендації щодо подальшої роботи в даному напрямку полягають у розширенні функціональності застосунку та підвищенні якості розпізнавання мовлення в різних умовах використання. Доцільним напрямом розвитку є проведення додаткових експериментальних досліджень продуктивності моделі на різних iOS-пристроях, оцінювання затримки live-транскрипції, аналіз енергоспоживання та порівняння якості роботи різних варіантів Whisper-моделей. Також перспективним є впровадження підтримки декількох мов, автоматичного визначення мови мовлення та покращення роботи із шумними аудіозаписами.

Подальший розвиток застосунку може передбачати додавання механізмів експорту нотаток у зовнішні формати, синхронізації між пристроями за умови збереження конфіденційності, розширеного редагування транскрипцій та категоризації записів. Окремим перспективним напрямом є вдосконалення AI-обробки тексту, зокрема додавання персоналізованих шаблонів перетворення нотаток, автоматичного формування задач, тез, підсумків зустрічей та структурованих документів. Отже, розроблений застосунок може бути основою для подальших досліджень у сфері on-device AI, мобільного розпізнавання мовлення та інтелектуальної обробки користувацьких текстових даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Radford A., Kim J. W., Xu T. та ін. Whisper: Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv.* 2022. URL: https://arxiv.org/abs/2212.04356 (date of access: 15.02.2026).

2. Apple Inc. Core ML Framework Documentation. *Apple Developer Documentation.* URL: https://developer.apple.com/documentation/coreml (date of access: 03.02.2026).

3. Apple Inc. Speech Framework Documentation. *Apple Developer Documentation.* URL: https://developer.apple.com/documentation/speech (date of access: 05.02.2026).

4. Apple Inc. AVFoundation Framework Documentation. *Apple Developer Documentation.* URL: https://developer.apple.com/documentation/avfoundation (date of access: 07.02.2026).

5. ONNX Runtime. CoreML Execution Provider. *ONNX Runtime Documentation.* URL: https://onnxruntime.ai/docs/execution-providers/CoreML-ExecutionProvider.html (date of access: 10.02.2026).

6. Hannun A. Deep Speech: Scaling up end-to-end speech recognition. *arXiv.* 2014. URL: https://arxiv.org/abs/1412.5567 (date of access: 12.02.2026).

7. Graves A., Mohamed A., Hinton G. Speech Recognition with Deep Recurrent Neural Networks. *arXiv.* 2013. URL: https://arxiv.org/abs/1303.5778 (date of access: 14.02.2026).

8. Vaswani A., Shazeer N., Parmar N. та ін. Attention Is All You Need. *arXiv.* 2017. URL: https://arxiv.org/abs/1706.03762 (date of access: 16.02.2026).

9. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. *Stanford University.* 2023. URL: https://web.stanford.edu/~jurafsky/slp3/ (date of access: 18.02.2026).

10. OpenAI. Whisper GitHub Repository. *GitHub.* URL: https://github.com/openai/whisper (date of access: 20.02.2026).

11. ggml Project. Efficient Inference of ML Models. *GitHub.* URL: https://github.com/ggerganov/ggml (date of access: 22.02.2026).

12. TensorFlow Lite. On-device Machine Learning. *TensorFlow Documentation.* URL: https://www.tensorflow.org/lite (date of access: 25.02.2026).

13. PyTorch Mobile. Machine Learning on Mobile Devices. *PyTorch Documentation.* URL: https://pytorch.org/mobile/ (date of access: 27.02.2026).

14. Tveit A., Morland T., Rost T. B. DeepLearningKit – GPU Optimized Deep Learning Framework for iOS, OS X and tvOS. *GitHub.* URL: https://github.com/DeepLearningKit/DeepLearningKit (date of access: 28.02.2026).

15. Chan W., Jaitly N., Le Q., Vinyals O. Listen, Attend and Spell. *arXiv.* 2016. URL: https://arxiv.org/abs/1508.01211 (date of access: 01.03.2026).

16. Baevski A., Zhou Y., Mohamed A., Auli M. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *arXiv.* 2020. URL: https://arxiv.org/abs/2006.11477 (date of access: 05.03.2026).

17. NVIDIA. Deep Learning for Speech Recognition. *NVIDIA Developer.* URL: https://developer.nvidia.com/deep-learning (date of access: 08.03.2026).

18. Brown T. B., Mann B., Ryder N. та ін. Language Models are Few-Shot Learners. *arXiv.* 2020. URL: https://arxiv.org/abs/2005.14165 (date of access: 12.03.2026).

19. Vajjala S., Majumder B., Gupta A., Surana H. Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems. Sebastopol: O'Reilly Media, 2020.

20. Apple Inc. Machine Learning and AI Technologies Overview. *Apple Developer.* URL: https://developer.apple.com/machine-learning/ (date of access: 18.03.2026).