МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЧНОГО АНАЛІЗУ ЕМОЦІЙ У ГОЛОСІ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.М. Мальцев
(ініціали та прізвище)
Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Горбенко В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Мальцеву Максиму Максимовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку для автоматичного аналізу емоцій у голосі методами машинного навчання

керівник роботи Горбенко Віталій Іванович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.
3. Комплект нормативних документів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження та аналіз існуючих підходів та методів автоматичного розпізнавання емоцій у голосі.
2. Розробка та навчання моделі машинного навчання для класифікації емоцій.
3. Розробка архітектури та реалізація вебзастосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Затвердження теми, завдання на кваліфікаційну роботу. Розробка детального календарного плану.	13.01.2025	
2.	Поглиблений аналіз літератури, існуючих рішень та методів розпізнавання емоцій.	03.02.2025	
3.	Проєктування архітектури системи: вибір технологій, розробка архітектури backend та frontend, API.	24.02.2025	
4.	Підготовка та попередня обробка даних. Розробка прототипу моделі машинного навчання.	17.03.2025	
5.	Розробка основної функціональності серверної частини.	07.04.2025	
6.	Розробка основної функціональності клієнтської частини.	12.05.2025	
7.	Підготовка презентації.	19.05.2025	
8.	Оформлення та нормоконтроль кваліфікаційної роботи бакалавра.	26.05.2025	
9.	Захист кваліфікаційної роботи.	18.06.2025	

Студент _____ М.М. Мальцев
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебзастосунку для автоматичного аналізу емоцій у голосі методами машинного навчання»: 64 с., 11 рис., 5 табл., 24 джерела, 1 додаток.

ГЛИБОКЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ ЕМОЦІЙ, ОБРОБКА АУДІОСИГНАЛІВ, РОЗПІЗНАВАННЯ ЕМОЦІЙ, FASTAPI, RAVDESS, TORCH.

Об’єкт дослідження – процес автоматичного аналізу та розпізнавання емоційного стану людини за акустичними характеристиками її голосу з використанням методів машинного навчання.

Мета роботи: розробка та реалізація функціональності вебзастосунку для здійснення автоматичного аналізу емоцій у голосі на основі попередньо навченої моделі машинного навчання.

Методи дослідження – теоретичний аналіз науково-технічної літератури, методи обробки цифрових аудіосигналів, методи машинного навчання, методи проєктування та розробки програмного забезпечення, включаючи клієнт-серверну архітектуру (REST API).

У роботі показано можливість розпізнавання емоцій у голосі за допомогою згорткових нейронних мереж. Для цього було застосовано стек технологій Python, FastAPI, PyTorch для розробки моделі та серверної частини, а HTML, CSS, JavaScript, Bootstrap – для клієнтської. У роботі розроблено архітектуру вебзастосунку з REST API та реалізовано систему з функціональністю аналізу аудіо різної тривалості, візуалізації спектрограм, динаміки емоцій та історії аналізів.

SUMMARY

Bachelor’s qualifying paper “Developing a Web Application for Automatic Emotion Analysis in the Voice by Machine Learning Methods”: 64 pages, 11 figures, 5 tables, 24 references, 1 supplement.

AUDIO SIGNAL PROCESSING, DEEP LEARNING, EMOTION CLASSIFICATION, EMOTION RECOGNITION, FASTAPI, RAVDESS, TORCH.

The object of the study is process of automatic analysis and recognition of a person’s emotional state by the acoustic characteristics of their voice using machine learning methods, and the development of corresponding software in the form of a web application.

The aim of the study is development and implementation of a functional web application for performing automatic analysis of emotions in voice based on a pre-trained machine learning mode.

The method of research are theoretical analysis of scientific and technical literature, methods of digital audio signal processing, machine learning methods, software design and development methods, including client-server architecture (REST API).

The thesis demonstrates the capability of recognizing emotions in voice using convolutional neural networks. For this purpose, a technology stack including Python, FastAPI, and PyTorch was applied for model and backend development, while HTML, CSS, JavaScript, and Bootstrap were used for the client-side. The work involved developing a web application architecture with a REST API and implementing a system with functionality for analyzing audio of varying durations, visualizing spectrograms, emotion dynamics, and analysis history.

ЗМІСТ

Завдання на кваліфікаційну роботу.....2
Реферат 4
Summary 5
Вступ..... 8
1 Теоретичні основи та аналіз методів автоматичного розпізнавання емоцій в голосі 10
1.1 Емоції людини: психофізіологічні аспекти та їх прояви у мовленні. 10
1.2 Огляд існуючих підходів та систем для автоматичного розпізнавання емоцій у голосі (SER) 12
1.3 Акустичні ознаки мовлення для ідентифікації емоцій 15
1.4 Методи машинного навчання у задачах класифікації емоцій..... 17
1.5 Характеристика та порівняльний аналіз наборів даних для навчання моделей SER 20
1.6 Висновки до розділу 21
2 Проєктування та розробка вебзастосунку для аналізу емоцій в голосі..... 23
2.1 Обґрунтування вибору технологічного стеку для реалізації системи API 23
2.2 Архітектура вебзастосунку: проєктування клієнт-серверної взаємодії та API..... 25
2.3 Проєктування моделі машинного навчання для розпізнавання емоцій 29
2.4 Проєктування користувацького інтерфейсу (UI/UX) вебзастосунку 32
2.5 Висновки до розділу 35
3 Реалізація, тестування та аналіз результатів роботи вебзастосунку..... 36
3.1 Реалізація серверної частини (Backend) на FastAPI 36
3.2 Реалізація клієнтської частини 38
3.3 Процес навчання, валідації та тестування моделі машинного навчання 40

3.4 Тестування функціональності вебзастосунку 43
3.5 Аналіз результатів роботи системи розпізнавання емоцій та обговорення перспектив її розвитку 45
3.6 Висновки до розділу 48
Висновки 49
Перелік посилань 52
Додаток А Програмна реалізація API 54

ВСТУП

В епоху цифрової трансформації, питання ефективної взаємодії людини з комп’ютерними системами набуває дедалі більшої значущості. Голосові інтерфейси стали невід’ємною частиною сучасного технологічного ландшафту, проте для справді природної комунікації машинам важливо не лише розпізнавати слова, а й розуміти емоційний контекст мовлення. Емоції є ключовим елементом людського спілкування, передаючи нюанси намірів та стану мовця.

Розпізнавання емоцій у голосі (Speech Emotion Recognition – SER) перетворилося з вузькоспеціалізованої задачі машинного навчання на важливу компоненту взаємодії «людина-комп’ютер». Ця багатопрофільна галузь досліджень останнім часом активно розвивається, що зумовлено зростанням попиту на інтелектуальні системи у сферах обслуговування клієнтів, психологічної підтримки, освітніх технологій та розважальної індустрії. Автоматичне розпізнавання емоцій за голосом є актуальною науково-технічною задачею, вирішення якої може суттєво покращити якість цифрових сервісів та відкрити нові можливості для їх застосування.

Хоча для людини визначення емоційного стану співрозмовника є відносно простим завданням, для комп’ютерних систем воно залишається нетривіальним. Сучасні системи розпізнавання мови та ідентифікації за голосом досягли значного прогресу, проте інтеграція функціоналу аналізу емоцій дозволяє перейти на новий рівень розуміння – аналізувати не лише хто і що сказав, а й як це було сказано. Емоції мають свої специфічні акустичні ознаки, закодовані в мовному сигналі, дослідження яких є ключовим для розробки ефективних алгоритмів SER. Зі зростанням популярності методів машинного навчання, зокрема глибоких нейронних мереж, з’являються нові можливості для вирішення цієї задачі, хоча низка проблем, пов’язаних із створенням доступних та зручних у використанні інструментів, залишається відкритою.

Актуальність теми даної кваліфікаційної роботи зумовлена потребою у розробці практично орієнтованих вебзастосунків, які б дозволяли широкому колу користувачів здійснювати аналіз емоцій у голосі без спеціалізованих знань, використовуючи сучасні досягнення в галузі машинного навчання.

Об’єкт дослідження – процес автоматичного аналізу та розпізнавання емоційного стану людини за акустичними характеристиками її голосу з використанням методів машинного навчання та розробка відповідного програмного забезпечення у вигляді вебзастосунку.

Для досягнення поставленої мети визначено такі завдання дослідження:

– проаналізувати сучасні підходи до розпізнавання емоцій у голосі та технології веброзробки;
– обґрунтувати вибір технологічного стеку: Python, FastAPI, torch, sklearn, librosa для серверної частини, MediaRecorder API для клієнтської [20];
– підготувати дані (RAVDESS) та розробити модель машинного навчання для класифікації емоцій;
– спроєктувати та реалізувати архітектуру та функціонал вебзастосунку.

Практичне значення отриманих результатів полягає у створенні доступного вебінструменту для аналізу емоцій, який може знайти застосування в психологічних дослідженнях, освітніх цілях, покращенні клієнтського сервісу та як основа для подальших розробок.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатку. Перший розділ присвячено аналізу предметної області та огляду методів розпізнавання емоцій. У другому розділі детально описано проєктування вебзастосунку та моделі машинного навчання. Третій розділ охоплює реалізацію, тестування системи та аналіз отриманих результатів.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ МЕТОДІВ АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ ЕМОЦІЙ В ГОЛОСІ

1.1 Емоції людини: психофізіологічні аспекти та їх прояви у мовленні

Емоції є фундаментальною складовою людського досвіду, що відіграють ключову роль у пізнанні, прийнятті рішень, соціальній взаємодії та загальному добробуті. З наукової точки зору, емоція – це складний психофізіологічний процес, що відображає суб'єктивне ставлення індивіда до існуючих або можливих ситуацій та об'єктивного світу. Емоції включають когнітивну оцінку, фізіологічні зміни, експресивну поведінку та суб'єктивне переживання [1]. Розуміння природи емоцій є передумовою для розробки ефективних систем їх автоматичного розпізнавання.

Існує декілька основних теоретичних підходів до класифікації емоцій. Одним із найвпливовіших є підхід, що базується на концепції базових емоцій. Пол Екман та його колеги ідентифікували шість (або більше, залежно від класифікації) універсальних емоцій, які, як вважається, мають чіткі та вроджені патерни вираження, зокрема мімічні та вокальні: радість, сум, гнів, страх, огида та здивування [2]. Ці емоції вважаються еволюційно зумовленими та культурно незалежними. Інший підхід пропонує Роберт Плутчик у своїй «колесі емоцій», де він виділяє вісім базових біполярних емоцій: радість – сум, довіра – огида, страх – гнів, здивування – очікування. Комбінації цих базових емоцій можуть утворювати складніші емоційні стани [2].

Альтернативою дискретним моделям є дименсійні моделі емоцій. Вони описують емоції не як окремі категорії, а як точки у багатовимірному просторі. Найпопулярнішою є двовимірна модель Рассела, яка описує емоції за двома осями: валентність (valence – від позитивного до негативного, задоволення – незадоволення) та збудження (arousal – від спокійного до збудженого,

активація – деактивація) [3]. Наприклад, радість характеризується високою валентністю та високим збудженням, тоді як сум – низькою валентністю та низьким збудженням. Іноді додають третій вимір, такий як домінантність (dominance – від підпорядкованого до домінантного) або контроль. Дименсійні моделі дозволяють описувати тонші відтінки емоційних станів та переходи між ними.

Прояви емоцій у мовленні надзвичайно різноманітні й охоплюють численні аспекти голосоутворення та артикуляції. Емоційний стан людини безпосередньо впливає на м’язову напругу, характер дихання та контроль над голосовими зв’язками, що й зумовлює ці зміни. Серед ключових акустичних характеристик, що варіюються під впливом емоцій, виділяють просодичні та спектральні особливості.

До просодичних характеристик належить, насамперед, частота основного тону (F0), або висота голосу, яка є однією з найважливіших ознак. Так, гнів, радість та страх часто супроводжуються підвищенням середнього значення F0 та розширенням її діапазону, тоді як для суму характерне зниження F0 та звуження діапазону [3]. Інтенсивність, або гучність голосу, також є показовою: висока інтенсивність часто асоціюється з гнівом, радістю чи страхом, а низька – із сумом. Темп мовлення може значно варіюватися: швидкий темп вказує на радість, страх або загальне збудження, тоді як повільний – на сум чи втому. Не менш важливими є ритм та інтонаційні контури, оскільки емоції впливають на мелодику мовлення, тривалість пауз та розстановку наголосів; наприклад, монотонне мовлення часто пов’язане з сумом або нудьгою.

Спектральні характеристики відображають зміни в енергетичному складі голосу. Емоції можуть змінювати розподіл енергії в спектрі, впливаючи на баланс між низькочастотними та високочастотними компонентами, що проявляється, наприклад, у «дзвінкому» голосі при радості або «глухому» при сумі. Формантні частоти, хоча й переважно визначають фонетичний зміст, також можуть демонструвати чутливість до емоційного стану через зміни у формі резонаторних порожнин рота та носа [3].

Окремо варто виділити параметри якості голосу (Voice Quality). Такі явища, як дрижання (Jitter – варіації частоти основного тону) та мерехтіння (Shimmer – варіації амплітуди), можуть вказувати на стрес, страх, гнів або сум, що пов’язані з нестабільністю роботи голосових зв’язок. Співвідношення гармонік до шуму (HNR) також є інформативним: низький HNR (більше шуму, «хрипкий» голос) може асоціюватися з сумом або гнівом, тоді як високий HNR (чистий голос) – зі спокоєм або радістю. Такі характеристики, як наявність придиху (breathiness), різкість (harshness) чи напруженість (tenseness) голосу, також змінюються залежно від емоційного стану.

Глибоке розуміння зв’язку між психофізіологічним станом людини та акустичними характеристиками її голосу є критично важливим для розробки ефективних алгоритмів автоматичного розпізнавання емоцій. Дослідження в цій багатогранній галузі невпинно продовжуються, адже індивідуальні особливості мовця, культурний контекст та навіть зміст конкретної фрази можуть суттєво впливати на вокальне вираження емоцій, що створює певні виклики для створення універсальних та високоточних моделей.

1.2 Огляд існуючих підходів та систем для автоматичного розпізнавання емоцій у голосі (SER)

Автоматичне розпізнавання емоцій у голосі, відоме як Speech Emotion Recognition (SER), є міждисциплінарною галуззю на стику обробки сигналів, акустики, психології та штучного інтелекту. Основна мета SER – створення комп’ютерних систем, здатних ідентифікувати емоційний стан людини на основі характеристик її мовлення. Хоча перші спроби пов’язати акустичні параметри з емоціями з’явилися ще в середині XX століття, справді стрімкий розвиток цієї сфери розпочався лише з появою потужних обчислювальних ресурсів та значним прогресом у методах машинного навчання наприкінці XX – на початку XXI століття [4].

Стандартна система SER зазвичай включає декілька ключових етапів обробки. Спочатку відбувається попередня обробка мовленнєвого сигналу, що передбачає видалення шумів (денеізінг), сегментацію мовлення для виділення активних ділянок голосу та усунення пауз, а іноді й нормалізацію самого сигналу. Якість цього етапу суттєво впливає на точність подальшого аналізу.

Наступним кроком є виділення ознак (Feature Extraction), під час якого з обробленого сигналу витягуються акустичні характеристики, що потенційно несуть інформацію про емоційний стан мовця. Правильний вибір ознак є одним із найважливіших завдань у SER. Оскільки кількість виділених ознак може бути дуже великою, часто застосовують методи для відбору та/або трансформації ознак (Feature Selection/Transformation) з метою зменшення розмірності простору та усунення надлишкових або неінформативних характеристик. Це може включати відбір найважливіших ознак за статистичними критеріями або їх трансформацію, наприклад, за допомогою методу головних компонент (PCA).

Далі, на основі обраного набору ознак, відбувається класифікація емоцій (Emotion Classification), де модель машинного навчання, навчена на розміченому наборі даних, розпізнає різні емоційні стани.

Завершальним етапом може бути постобробка та прийняття рішення, де результати класифікації додатково обробляються, наприклад, шляхом згладжування послідовності передбачених емоцій у часі або врахування контекстуальної інформації.

Існуючі підходи до SER можна класифікувати за кількома основними критеріями. За залежністю від мовця розрізняють:

- мовцезалежні системи, що навчаються на даних одного мовця і демонструють високу точність для нього, але погано узагальнюються;
- мовце-незалежні системи, навчені на даних багатьох мовців і призначені для розпізнавання емоцій будь-якої людини, що є складнішим, але більш практичним завданням;

– адаптивні до мовця системи, які є компромісом, де загальна модель адаптується до особливостей нового мовця.

За типом розпізнаваних емоцій системи поділяються на: дискретні (категоріальні), що розпізнають емоції як окремі категорії (наприклад, радість, сум, гнів), і вони є найпоширенішими; та дименсійні системи, що оцінюють емоції за безперервними шкалами, такими як валентність-збудження-домінантність, дозволяючи описувати тонші відтінки емоцій, але є складнішими в реалізації.

Щодо характеру мовлення, існують системи, розроблені для акторського (зіграного) мовлення, де емоції чітко демонструються (наприклад, датасети RAVDESS, EMODB), що полегшує навчання моделей; та системи для природного (спонтанного) мовлення, навчені на даних з реальних життєвих ситуацій. Розпізнавання емоцій у спонтанному мовленні є значно складнішим завданням через меншу виразність емоцій, їх можливе змішування та наявність шумів.

Сфери застосування систем SER надзвичайно широкі й охоплюють: покращення взаємодії «людина-комп’ютер» через створення більш емпатичних віртуальних асистентів та роботів; моніторинг емоційного стану в кол-центрах для оцінки якості обслуговування; діагностику та моніторинг психічних розладів у медицині та психології; детектування стресу або агресії в системах безпеки; а також створення інтерактивних ігор та розважальних додатків.

Незважаючи на значний прогрес у цій галузі, залишається низка викликів, що потребують подальших досліджень. До них належать забезпечення надійності розпізнавання в умовах зашумленого середовища, узагальнення моделей на різні мови та культурні особливості вираження емоцій, розпізнавання тонких та змішаних емоційних станів, а також вирішення етичних аспектів, пов’язаних з використанням таких технологій. Розробка доступних вебзастосунків для SER, подібних до представленого у даній кваліфікаційній роботі, сприяє популяризації цих технологій та розширенню кола користувачів, які можуть ними скористатися.

1.3 Акустичні ознаки мовлення для ідентифікації емоцій

Вибір та ефективне виділення акустичних ознак є фундаментальним завданням у побудові будь-якої системи автоматичного розпізнавання емоцій у голосі. Ці ознаки, по суті, є числовими дескрипторами, що кількісно характеризують різні аспекти мовленнєвого сигналу і несуть потенційну інформацію про емоційний стан мовця. Традиційно їх можна умовно поділити на кілька основних категорій, серед яких просодичні, спектральні, кепстральні та ознаки якості голосу [5].

Просодичні ознаки охоплюють так звані супрасегментні характеристики мовлення – інтонацію, ритм, наголос та темп, які накладаються на послідовність фонем і тісно пов’язані з емоційним забарвленням висловлювання. Однією з найважливіших просодичних ознак є частота основного тону (F0), що сприймається як висота голосу і відповідає найнижчій частоті вібрації голосових зв’язок. Різноманітні статистичні показники F0, такі як її середнє значення, медіана, стандартне відхилення, діапазон, а також характеристики самого контуру F0 (наприклад, нахил чи форма), є надзвичайно інформативними. Зокрема, високе середнє значення F0 та широкий діапазон часто асоціюються з радістю, гнівом або страхом, тоді як низьке середнє F0 та вузький діапазон частіше вказують на сум [4].

Інтенсивність, або гучність мовлення, також є важливою просодичною характеристикою, де високі значення часто пов’язані з гнівом та радістю, а низькі – із сумом. Тривалісні характеристики, що включають загальний темп мовлення (кількість слів чи складів за одиницю часу), тривалість окремих звуків та пауз, також відображають емоційний стан: швидкий темп може вказувати на збудження (радість, страх), а повільний темп і довгі паузи – на сум. Нарешті, ритм мовлення, що характеризує його часову структуру та патерни наголошених і ненаголошених складів, доповнює набір просодичних індикаторів емоцій. Для виділення цих ознак часто застосовують алгоритми автокореляції, методи на основі кепстрального аналізу або використовують спеціалізовані інструменти, такі як Praat чи бібліотека Librosa в Python.

Спектральні ознаки описують розподіл енергії мовленнєвого сигналу за різними частотами, відображаючи при цьому характеристики роботи голосового тракту, зокрема резонаторні властивості ротової та носової порожнин. Серед них важливе місце займають формантні частоти – піки в спектральній огинаючій сигналу, що відповідають резонансам голосового тракту і є ключовими для розрізнення голосних звуків. Хоча основна роль формант фонетична, їхні значення (особливо перших трьох – F1, F2, F3) та ширина смуги пропускання можуть незначно змінюватися під впливом емоцій через зміни у напруженні м’язів артикуляційного апарату [4]. Також аналізується спектральна енергія в різних частотних діапазонах. Наприклад, збільшення енергії у високочастотній області може бути пов’язане з гнівом або радістю. Спектральний нахил, що описує спад енергії спектру зі зростанням частоти, може вказувати на зміни в напруженості голосових зв’язок. До спектральних ознак належать і різні параметри, що описують форму спектральної огинаючої, такі як спектральний центроїд, спектральна дисперсія та спектральний потік.

Кепстральні ознаки отримуються в результаті оберненого перетворення Фур’є від логарифму амплітудного спектру сигналу. Ці коефіцієнти мають важливу властивість – вони добре розділяють інформацію про джерело збудження (голосові зв’язки) та характеристики голосового тракту. Найпопулярнішими та найефективнішими кепстральними ознаками в системах розпізнавання мови та емоцій є мел-частотні кепстральні коефіцієнти (MFCC) [6]. Процес їх отримання включає сегментацію сигналу, обчислення спектру потужності, фільтрацію за допомогою банку мел-фільтрів (що імітують сприйняття частоти людським вухом) та застосування дискретного косинусного перетворення.

Окрім згаданих, існують також ознаки якості голосу, які характеризують періодичність, шумність та інші аспекти роботи голосових зв’язок, такі як дрижання (Jitter), мерехтіння (Shimmer) та співвідношення гармонік до шуму (HNR), що також можуть змінюватися залежно від емоційного стану. Успішне

розпізнавання емоцій часто залежить від правильного вибору та комбінації різних типів цих акустичних ознак.

1.4 Методи машинного навчання у задачах класифікації емоцій

Після виділення інформативних акустичних ознак ключовим етапом у системах розпізнавання емоцій у голосі (SER) стає побудова ефективного класифікатора. Саме цей класифікатор, на основі отриманих ознак, присвоює мовленнєвому фрагменту відповідну емоційну мітку. Вибір методу машинного навчання має вирішальне значення, оскільки він значною мірою визначає точність та узагальнюючу здатність всієї системи. Історично в галузі SER застосовувалися різноманітні підходи, починаючи від класичних статистичних моделей і закінчуючи сучасними архітектурами глибоких нейронних мереж [7].

До широкого розповсюдження глибокого навчання, в задачах SER активно використовувалися так звані класичні, або традиційні, алгоритми машинного навчання. Характерною особливістю цих методів є те, що вони зазвичай потребують ретельного попереднього відбору та конструювання ознак (feature engineering) експертом. Серед найпопулярніших класичних класифікаторів у SER варто відзначити метод опорних векторів (Support Vector Machines – SVM). Його популярність зумовлена здатністю ефективно працювати з високорозмірними просторами ознак та хорошою продуктивністю навіть на відносно невеликих наборах даних [8]. SVM функціонує шляхом пошуку оптимальної гіперплощини, яка найкращим чином розділяє дані різних класів. Застосування різноманітних ядерних функцій, таких як лінійна, поліноміальна чи радіальна базисна функція (RBF), дозволяє SVM моделювати складні нелінійні залежності між ознаками та емоціями. Для задач багатокласової класифікації, якою є розпізнавання декількох емоцій, SVM зазвичай реалізується через стратегії «один проти всіх» або «один проти одного».

Іншим поширеним класичним методом є k-найближчих сусідів (k-Nearest Neighbors – k-NN). Це простий непараметричний алгоритм, який класифікує новий об’єкт на основі класу більшості серед його k найближчих сусідів у просторі ознак [8]. Важливими параметрами для k-NN є вибір метрики відстані (наприклад, Евклідова) та саме значення k. Однак, k-NN може бути чутливим до так званого «прокляття розмірності» при великій кількості ознак та вимагає ретельного масштабування даних. Також у SER застосовувалися гауссові сумішеві моделі (Gaussian Mixture Models – GMM). Це ймовірнісні моделі, які представляють розподіл даних кожного емоційного класу як суміш декількох гауссових розподілів [8]. GMM, що часто використовувалися для моделювання акустичних ознак у загальному розпізнаванні мови, продемонстрували свою ефективність і в контексті емоцій. Класифікація за допомогою GMM відбувається шляхом обчислення ймовірності приналежності об’єкта до кожного класу та вибору класу з максимальною ймовірністю. Варто згадати і приховані Марковські моделі (HMM), які добре підходять для моделювання послідовностей, якими є мовленнєві сигнали, та ансамблеві методи, такі як випадковий ліс чи градієнтний бустинг.

Останніми роками справжню революцію в багатьох галузях штучного інтелекту, включаючи SER, здійснили методи глибокого навчання (Deep Learning – DL). Головною перевагою глибоких нейронних мереж є їхня здатність до автоматичного вивчення ієрархічних представлень ознак (feature learning) безпосередньо з сирих або низькорівневих даних, таких як спектрограми або послідовності MFCC. Це значно зменшує потребу в трудомісткій ручній інженерії ознак, яка є типовою для класичних підходів [8].

Серед архітектур глибокого навчання, що довели свою ефективність у SER, особливе місце займають згорткові нейронні мережі (Convolutional Neural Networks – CNN). Спочатку розроблені для обробки зображень, CNN виявилися надзвичайно дієвими для аналізу спектрограм мовленнєвих сигналів, які можна розглядати як своєрідні двовимірні зображення, де осі

представляють час та частоту [8]. Згорткові шари CNN здатні автоматично виявляти важливі локальні патерни в спектрограмах (наприклад, зміни формант, гармонічну структуру), а шари підвибірки (пулінгу) зменшують розмірність даних та забезпечують певну інваріантність до невеликих зсувів цих патернів. Таким чином, CNN можуть ефективно вивчати просторово-частотні характеристики, що є ключовими для розрізнення емоцій.

Для обробки послідовних даних, якими є мовленнєві сигнали або виділені з них часові ряди ознак, природним вибором є рекурентні нейронні мережі (Recurrent Neural Networks – RNN). Особливістю RNN є наявність внутрішньої «пам’яті», що дозволяє мережі враховувати попередню інформацію при обробці поточного вхідного елемента. У контексті SER широкого застосування набули такі вдосконалені типи RNN, як мережі довгої короткочасної пам’яті (Long Short-Term Memory – LSTM). Вони спеціально розроблені для подолання проблеми зникаючого або вибухаючого градієнта, характерної для стандартних RNN, що дозволяє LSTM ефективно моделювати довготривалі залежності в послідовностях [8]. Це робить їх особливо корисними для аналізу часової динаміки емоційних проявів у мовленні. Спрощеною, але часто не менш ефективною альтернативою LSTM є керовані рекурентні блоки (Gated Recurrent Units – GRU), які мають меншу кількість параметрів [9]. Також популярними є гібридні архітектури, що поєднують CNN для виділення ознак та RNN для моделювання часових залежностей (CRNN), а також моделі з механізмами уваги та трансформери.

Вибір конкретної архітектури глибокого навчання завжди залежить від специфіки завдання, обсягу та типу доступних навчальних даних, а також наявних обчислювальних ресурсів. Моделі глибокого навчання, незважаючи на свою потужність, зазвичай вимагають значно більших обсягів даних для ефективного навчання та більших обчислювальних потужностей порівняно з класичними методами. Крім того, вони часто сприймаються як «чорні скриньки» через складність інтерпретації їхніх внутрішніх механізмів прийняття рішень.

У даній кваліфікаційній роботі, враховуючи обраний технологічний стек, що включає бібліотеки PyTorch та Scikit-learn, було зосереджено увагу на розробці та дослідженні моделі на основі глибоких нейронних мереж, а саме згорткової нейронної мережі для аналізу спектрограм, з метою досягнення високої точності класифікації емоцій на датасеті RAVDESS.

1.5 Характеристика та порівняльний аналіз наборів даних для навчання моделей SER

Якість та характеристики навчальних даних є критично важливими для успішної побудови систем автоматичного розпізнавання емоцій у голосі. Модель машинного навчання може бути настільки хорошою, наскільки хороші дані, на яких вона навчалася. В галузі SER існує низка публічно доступних та широко використовуваних датасетів, які відрізняються за мовою, типом емоцій, способом збору даних (акторські чи спонтанні), кількістю мовців та обсягом [9]. Детальна статистика аналізу різних наборів даних для навчання моделей SER наведена у таблиці 1.1 [22-24].

Таблиця 1.1 – Огляд популярних датасетів

Назва датасету	Мова	Тип мовлення	К-сть мовців	Основні емоції	Обсяг (приблизно)	Особливості
RAVDESS	Англійська	Акторське	24	Нейтр, спокій, радість, сум, гнів, страх, огида, здив.	~7300 файлів	Висока якість інтенсивності, спів
EMODB	Німецька	Акторське	10	Гнів, нудьга, огида, страх, радість	~535 файлів	Висока якість анотації, бенчмарк
SAVEE	Англійська	Акторське	4 (чол.)	Гнів, огида, страх, радість, сум, здив, нейтр.	480 файлів	Мультимодальний, мало мовців

Продовження таблиці 1.1

Назва датасету	Мова	Тип мовлення	К-сть мовців	Основні емоції	Обсяг (приблизно)	Особливості
TESS	Англійська	Акторське	2 (жін.)	Гнів, огида, страх, радість, сум, здив, нейтр.	2800 файлів	Багато слів, сильно мовце-залежний
IEMOCAP	Англійська	Діалоги	10	Різні диск дименсійні	~12 годин	Більш природні, дименсійні мітки, складний

В кваліфікаційній роботі використовувався датасет RAVDESS через його доступність, високу якість, чітко виражені емоції та достатню кількість мовців для навчання мовце-незалежної моделі початкового рівня [21]. За можливості, розглядається варіант доповнення RAVDESS іншими датасетами або застосування технік аугментації даних для покращення узагальнюючої здатності моделі.

1.6 Висновки до розділу

У першому розділі кваліфікаційної роботи було проведено комплексний аналіз теоретичних основ та сучасного стану проблеми автоматичного розпізнавання емоцій у голосі. Розглянуто психофізіологічну природу емоцій та особливості їх вокального вираження, що є фундаментальним для розуміння завдань SER. Визначено, що просодичні, спектральні та кепстральні характеристики мовлення, а також ознаки якості голосу, що несуть ключову інформацію про емоційний стан мовця.

Проаналізовано існуючі підходи до побудови систем SER, включаючи етапи попередньої обробки сигналу, виділення та відбору ознак, а також класифікації. Здійснено огляд класичних методів машинного навчання, таких як SVM, GMM, HMM, та сучасних підходів на основі глибокого навчання, зокрема згорткових та рекурентних нейронних мереж (CNN, LSTM/GRU), а

також комбінованих архітектур та моделей з механізмами уваги. Підкреслено переваги методів глибокого навчання у здатності автоматично вивчати релевантні ознаки, але також вказано на їх вимогливість до обсягів даних та обчислювальних ресурсів.

Особливу увагу приділено аналізу акустичних ознак, що використовуються для ідентифікації емоцій. Детально розглянуто такі важливі групи ознак, як частота основного тону, інтенсивність, мел-частотні кепстральні коефіцієнти (MFCC) та хрома-ознаки, а також методи їх виділення за допомогою сучасних інструментів, таких як бібліотека librosa.

Таким чином, перший розділ закладає необхідне теоретичне підґрунтя для подальшого проєктування та розробки вебзастосунку для автоматичного аналізу емоцій у голосі, окреслюючи ключові концепції, методи та інструменти, що будуть використані в наступних розділах роботи.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ЕМОЦІЙ В ГОЛОСІ

2.1 Обґрунтування вибору технологічного стеку для реалізації системи API

Успішна реалізація будь-якого програмного проєкту значною мірою залежить від правильно підібраного набору технологій. Для розробки вебзастосунку автоматичного аналізу емоцій у голосі було обрано стек інструментів, що оптимально поєднує можливості для наукомістких обчислень, розробки ефективного серверного API та створення зручного користувацького інтерфейсу. Ключовим критерієм вибору була синергія між компонентами стеку та їх відповідність специфіці поставлених завдань.

Серверна частина системи, що відповідає за обробку аудіо, роботу моделі машинного навчання та логіку API, реалізована на мові програмування Python. Цей вибір зумовлений потужною екосистемою Python для наукових досліджень та машинного навчання, що включає такі бібліотеки, як Torch (PyTorch) для глибокого навчання, Librosa для аналізу аудіосигналів та Scikit-learn для допоміжних завдань машинного навчання та оцінки моделей [13, 14]. PyTorch, зокрема, надав необхідну гнучкість для побудови та навчання кастомної архітектури згорткової нейронної мережі (SpectrogramCNN), яка використовується для класифікації емоцій на основі мел-спектрограм. Бібліотека Librosa стала незамінною для виділення цих спектрограм з аудіофайлів.

Для створення веб-API було обрано сучасний Python-фреймворк FastAPI. Його переваги, такі як висока продуктивність завдяки асинхронності, автоматична валідація даних за допомогою Pydantic та вбудована генерація інтерактивної документації (Swagger UI), суттєво прискорили процес розробки та тестування серверної частини. Асинхронна природа FastAPI також є важливою для потенційної обробки декількох запитів одночасно, хоча основне

навантаження в даному проєкті припадає на обчислення моделі. Для маніпуляцій з аудіофайлами, їх конвертації та сегментації на сервері використовується бібліотека Pydub, яка добре інтегрується в загальний Python-стек. Збереження та завантаження навчених артефактів, таких як модель, кодувальник міток (LabelEncoder) та параметри нормалізації, здійснюється за допомогою стандартного модуля Pickle. Для візуалізації спектрограм, що потім передаються на клієнт, та побудови графіків навчання використовується бібліотека Matplotlib. Запуск FastAPI застосунку забезпечується ASGI сервером Uvicorn. Детальний огляд технологій та їх призначень в проєкті наведено у таблиці 2.1.

Таблиця 2.1 – Технології серверної частини та їх призначення

Технологія/ Бібліотека	**Основне призначення в проєкті**
Python 3.0	Основна мова програмування, середовище для всіх компонентів.
FastAPI	Створення високопродуктивного REST API, валідація даних, автоматична документація.
Scikit-learn	Кодування міток (LabelEncoder), розділення даних, метрики оцінки моделі.
Librosa	Завантаження аудіо, виділення мел-спектрограм, візуалізація аудіо ознак.
Pydub	Конвертація аудіо форматів/параметрів, сегментація довгих аудіофайлів.
Matplotlib	Генерація зображень спектрограм для API та звіту [16].
Pickle	Серіалізація та збереження об’єктів Python (LabelEncoder, параметри нормалізації).
NumPy	Ефективна робота з числовими масивами (спектрограми, ознаки).
Uvicorn	ASGI сервер для запуску FastAPI застосунку.

Для розробки клієнтської частини було прийнято рішення використовувати класичний підхід, що базується на HTML5, CSS3 та JavaScript. Такий вибір забезпечує широку сумісність та відносну простоту реалізації для односторінкового застосунку з обмеженою кількістю інтерактивних елементів. Для прискорення розробки користувацького інтерфейсу та забезпечення його адаптивності було інтегровано фреймворк

Bootstrap 5 [17]. Він надав готові компоненти, такі як навігаційна панель, кнопки, форми, картки та модальні вікна, а також потужну систему сіток для коректного відображення на різних пристроях. Кастомні стилі, що доповнюють та модифікують Bootstrap, зберігаються у файлі css/style.css.

Взаємодія між клієнтською та серверною частинами здійснюється за допомогою HTTP-запитів, для формування яких на клієнті використовується бібліотека Axios. Вона спрощує роботу з асинхронними запитами та обробку відповідей від API [18].

Для візуалізації даних, зокрема розподілу ймовірностей емоцій та динаміки їх зміни у часі для довгих аудіофайлів, застосовується бібліотека Chart.js, що дозволяє створювати інтерактивні діаграми на основі HTML5 Canvas [19].

Потенційно, для реалізації функціоналу «Live-аналіз», може бути задіяний вбудований браузерний MediaRecorder API для запису аудіо з мікрофона користувача.

2.2 Архітектура вебзастосунку: проєктування клієнт-серверної взаємодії та API

Для забезпечення ефективної роботи та чіткого розподілу відповідальностей, вебзастосунок для аналізу емоцій у голосі було спроєктовано на основі класичної клієнт-серверної архітектури. Такий підхід передбачає наявність двох основних компонентів: клієнтської частини (frontend), що працює у веббраузері користувача, та серверної частини (backend), що виконує основні обчислення та обробку даних. Взаємодія між цими компонентами здійснюється через чітко визначений програмний інтерфейс застосунку (API), реалізований за принципами REST.

Клієнтська частина (Frontend), розроблена з використанням HTML, CSS та JavaScript з фреймворком Bootstrap, відповідає за користувацький інтерфейс та взаємодію з користувачем. Її основні функції включають:

– надання інтерфейсу для завантаження аудіофайлів у форматі .wav;
– забезпечення можливості запису аудіо в реальному часі за допомогою мікрофона;
– відправку аудіоданих на сервер для аналізу;
– отримання та візуалізацію результатів аналізу, включаючи передбачену емоцію, її ймовірнісний розподіл, спектрограми та іншу релевантну інформацію;
– відображення історії попередніх аналізів;
– навігацію між різними функціональними розділами застосунку.

Серверна частина (Backend), реалізована на Python з використанням фреймворку FastAPI, є ядром системи [10]. Вона виконує:

– прийом HTTP-запитів від клієнтської частини;
– обробку та валідацію вхідних даних, зокрема, аудіофайлів;
– перетворення аудіофайлів у формат, придатний для аналізу моделлю машинного навчання та генерує мел-спектрограми;
– нормалізацію спектрограм на основі параметрів, отриманих під час навчання моделі;
– застосування навченої моделі SpectrogramCNN для передбачення емоційного стану;
– обробку довгих аудіофайлів шляхом їх сегментації та аналізу кожного сегмента;
– формування відповіді з результатами аналізу у форматі JSON;
– збереження інформації про аналізи в історію (у файл analysis_history.json);
– надання доступу до історії аналізів.

Взаємодія між клієнтом та сервером відбувається асинхронно. Клієнт відправляє запит на сервер (наприклад, з аудіофайлом для аналізу) і не блокує інтерфейс користувача, очікуючи на відповідь. Після завершення обробки на сервері, результат повертається клієнту, який динамічно оновлює сторінку для відображення отриманих даних.

Програмний інтерфейс застосунку (API) спроєктовано з дотриманням принципів RESTful. Він визначає набір ендпоінтів (URL-адрес), через які клієнт може взаємодіяти з сервером, типи HTTP-методів (GET, POST), формати даних запитів та відповідей.

Основним програмним інтерфейсом системи для аналізу емоційного забарвлення аудіо є ендпоінт POST /api/audio-analyze. Він призначений для обробки завантажених аудіофайлів з метою визначення емоцій. Взаємодія з ним здійснюється методом POST, при цьому тіло запиту у форматі multipart/form-data повинно містити аудіофайл, переданий під ключем file.

У разі успішної обробки (статус 200 OK) сервер повертає JSON-об’єкт з результатами аналізу, структура якого варіюється залежно від тривалості аудіофайлу.

Для коротких аудіозаписів відповідь включає передбачену емоцію, її символьний ключ, рівень впевненості моделі, словник імовірностей для всіх розпізнаваних класів емоцій, а також base64-кодоване зображення спектрограми аналізованого аудіо та шлях до еталонної спектрограми для передбаченої емоції.

Для аудіофайлів значної тривалості відповідь містить прапорець is_long_audio: true, ідентифікує переважаючу емоцію та її ключ, а також надає масив результатів аналізу по окремих сегментах, де кожен сегмент характеризується часовою позначкою, визначеною емоцією, впевненістю та розподілом імовірностей.

Додатково повертається base64-кодоване зображення спектрограми першого проаналізованого сегмента та шлях до прикладу спектрограми, що відповідає переважаючій емоції.

У випадку виникнення помилок (наприклад, невалідний формат файлу – 400 Bad Request, або внутрішня помилка сервера – 500 Internal Server Error) API повертає JSON-об’єкт, що містить відповідний статус помилки та текстовий опис проблеми.

Для повного розуміння алгоритму, за яким відбувається робота сторінки «Аналіз аудіофайлу» створено діаграму послідовності на рисунку 2.1.

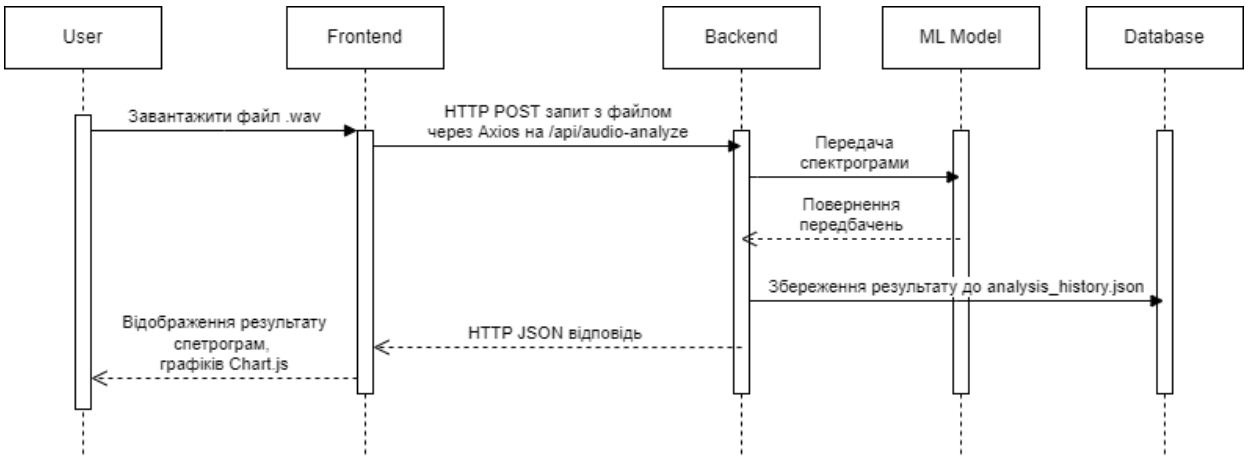


Рисунок 2.1 – Сценарій «Аналіз аудіофайлу»

Іншим важливим компонентом API є ендпоінт GET /api/history, який забезпечує доступ до історії раніше виконаних аналізів. Даний ендпоінт викликається методом GET і не вимагає передачі тіла запиту. У відповідь на успішний запит (статус 200 OK) сервер повертає JSON-масив, де кожен елемент є об’єктом, що репрезентує окремий запис з історії. Такий запис агрегує ключову інформацію про аналіз, включаючи час його проведення, назву обробленого файлу, визначену емоцію (або переважаючу для довготривалих аудіозаписів), тривалість файлу та інші супутні метадані. У випадку помилки клієнт отримує JSON-об’єкт з відповідним кодом статусу та описом проблеми. Для повного розуміння алгоритму, за яким відбувається робота сторінки «Аналіз аудіофайлу» створено діаграму послідовності на рисунку 2.2.

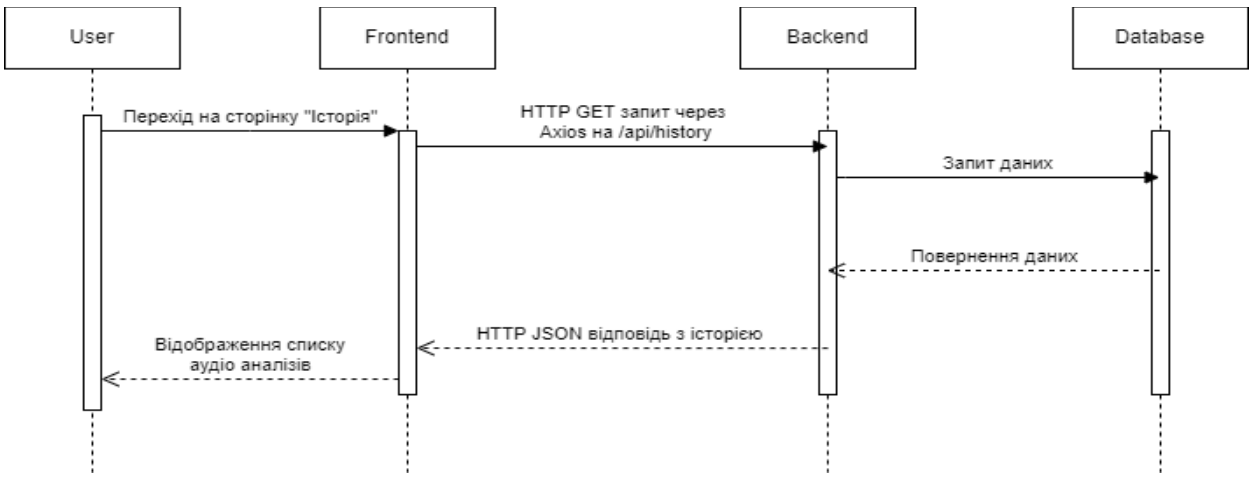


Рисунок 2.2 – Сценарій «Запит історії аналізів»

2.3 Проєктування моделі машинного навчання для розпізнавання емоцій

Центральним компонентом розробленого вебзастосунку є модель машинного навчання, відповідальна за безпосереднє розпізнавання емоцій у голосових сигналах. Вибір архітектури моделі та підхід до її навчання є ключовими факторами, що визначають точність та ефективність всієї системи. Враховуючи сучасні тенденції в галузі обробки аудіо та розпізнавання образів, для даного проєкту було обрано підхід на основі глибокого навчання, зокрема використання згорткових нейронних мереж (CNN) для аналізу мел-спектрограм.

На початкових етапах дослідження розглядалися різні підходи, включаючи класичні методи машинного навчання на основі акустичних ознак низького рівня (наприклад, MFCC, просодичні характеристики) та простіші нейромережеві архітектури типу багатошарового персептрону (MLP). Однак, для досягнення вищої точності та кращого узагальнення, було прийнято рішення використовувати мел-спектрограми як вхідні дані для моделі.

Мел-спектрограми є двовимірним представленням аудіосигналу, де одна вісь відповідає часу, а інша – мел-шкалі частот, що імітує людське сприйняття висоти звуку. Енергія сигналу на перетині часу та частоти відображається інтенсивністю кольору або значенням. Таке представлення дозволяє візуалізувати часово-частотні характеристики мовлення, які є важливими для розрізнення емоцій (наприклад, зміни основного тону, формант, розподілу енергії). Перевагою використання спектрограм є те, що модель CNN може автоматично вивчати релевантні локальні патерни та ознаки безпосередньо з «сирого» представлення, на відміну від підходів, що потребують ручної інженерії ознак.

Для класифікації емоцій на основі мел-спектрограм було розроблено згорткову нейронну мережу SpectrogramCNN, програмну реалізацію якої, зображено на рисунку 2.3.

```
class SpectrogramCNN(nn.Module):
    def __init__(self, num_mels, time_steps, num_classes, dropout_rate=0.4):
        super(SpectrogramCNN, self).__init__()
        self.conv_block1 = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding='same'), nn.ReLU(),
            nn.BatchNorm2d(32), nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.conv_block4 = nn.Sequential(
            nn.Conv2d(128, 128, kernel_size=3, padding='same'), nn.ReLU(),
            nn.BatchNorm2d(128), nn.MaxPool2d(kernel_size=(2,4)) # Асиметричний пулінг
        )
        # Динамічний розрахунок розміру для Flattened шару
        with torch.no_grad(): # Не потрібні градієнти для цього розрахунку
            dummy_input = torch.randn(1, 1, num_mels, time_steps)
            dummy_out = self._forward_conv_blocks(dummy_input)
            self.flattened_size = dummy_out.view(1, -1).size(1)
        self.fc_block = nn.Sequential(
            nn.Flatten(),
            nn.Linear(self.flattened_size, 256), nn.ReLU(),
            nn.Dropout(dropout_rate),
            nn.Linear(256, num_classes)
        )
```

Рисунок 2.3 – Визначення класу моделі SpectrogramCNN на PyTorch (ключові фрагменти)

Для навчання моделі використовувався датасет RAVDESS, з якого було відібрано аудіозаписи, що відповідають п'яти цільовим емоціям: спокій, радість, сум, злість та страх. Усі аудіофайли були перетворені на мел-спектрограми фіксованого розміру (N_MELS x FIXED_TIME_STEPS) та нормалізовані. Навчання моделі відбувалось за допомоги трьох вибірок, наведених у таблиці 2.2.

Таблиця 2.2 – Вибірки для моделі

Тренувальна (Training set)	70% даних, використовується для безпосереднього навчання моделі та оновлення її ваг.
Валідаційна (Validation set)	15% даних, використовується для моніторингу процесу навчання, виявлення перенавчання, налаштування гіперпараметрів (через ReduceLROnPlateau) та вибору найкращої моделі (через EarlyStopping). Модель не навчається на цих даних.
Тестова (Test set)	15% даних, використовується лише один раз після завершення навчання для фінальної, незалежної оцінки узагальнюючої здатності навченої моделі.

Для роботи з даними в PyTorch було створено кастомний клас RavdessSpectrogramDataset, що успадковується від torch.utils.data.Dataset [12]. Цей клас відповідає за завантаження спектрограми за шляхом до файлу, її нормалізацію та застосування аугментації (для тренувальної вибірки). Аугментація включала випадкове маскування часових та частотних смуг на спектрограмі, що допомагає зробити модель більш стійкою та покращити її узагальнення.

Модель непогано справляється з розпізнаванням “angry”, “calm”, “sad”, але плутає “happy” з “fearful”. Це відбувається через те, що актори відіграли ці емоції дуже подібно, а також аудіозаписи з цими емоціями були короткі за тривалістю. Найкраще розпізнається емоція “angry” – 27 правильних із 29 випадків (всього 2 помилки), це можна побачити на рисунку 2.4.

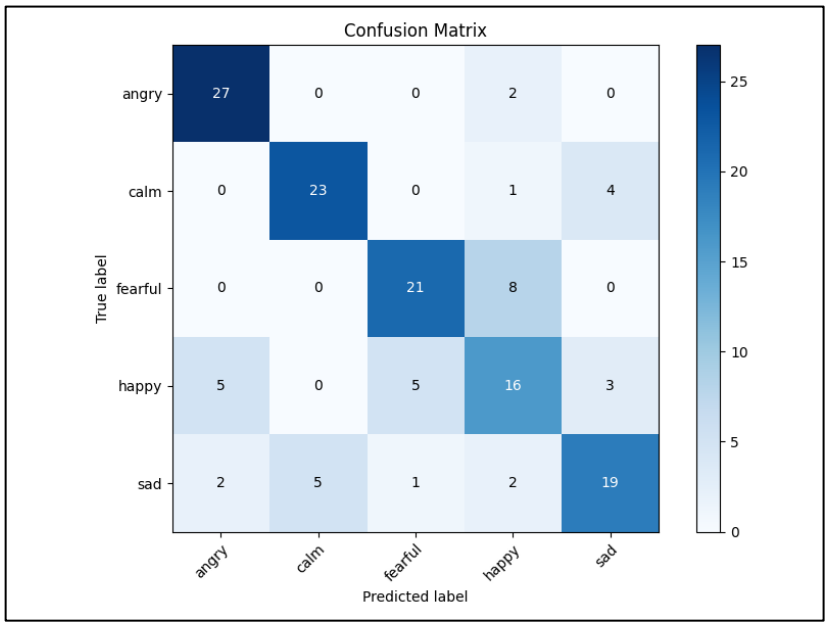


Рисунок 2.4 – Матриця помилок на тестовому наборі

Під час роботи було покращено баланс класів у даних, але це не привело до змін. Потенційно можна було зробити перенесення навчання на інший набір даних. Загалом планувалось навчання моделі на 150 епохах, але після 40-ої епохи точність моделі не покращувалась (ще протягом двадцяти епох), тому навчання моделі було примусово зупинено зі збереженням найкращих показників, які зображено на рисунку 2.5.

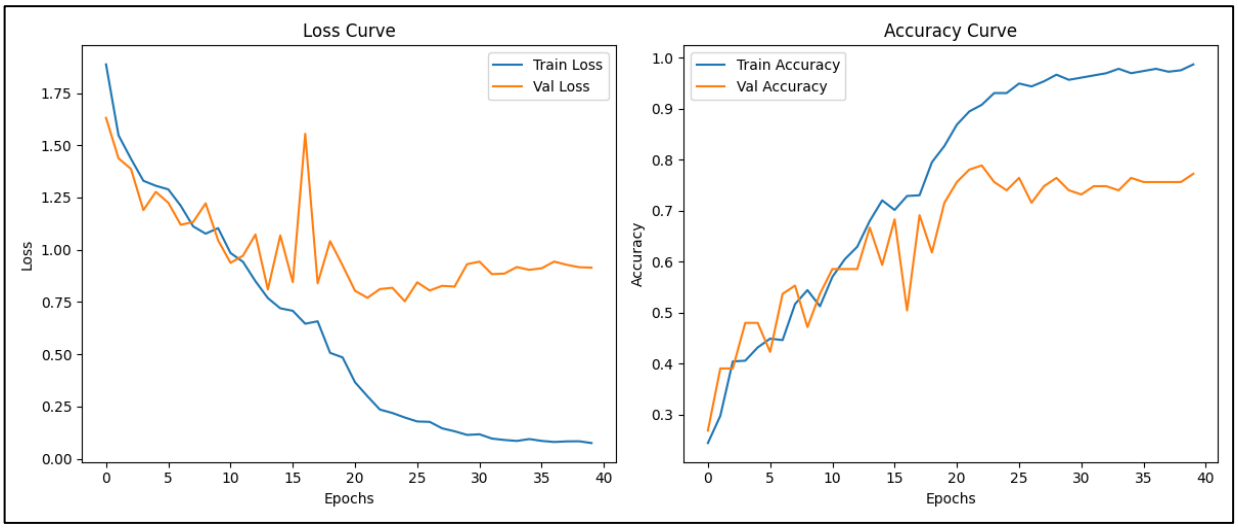


Рисунок 2.5 – Залежності змінень втрат та точності під час навчання моделі

2.4 Проєктування користувацького інтерфейсу (UI/UX) вебзастосунку

Ефективність та зручність використання будь-якого програмного продукту значною мірою визначаються якістю його користувацького інтерфейсу (UI) та загальним досвідом взаємодії (UX). При проєктуванні вебзастосунку для аналізу емоцій у голосі головна увага приділялася створенню інтуїтивно зрозумілого, функціонального та візуально приємного середовища для користувача. Оскільки цільова аудиторія може включати як технічних спеціалістів, так і звичайних користувачів, зацікавлених у можливостях технології, інтерфейс мав бути простим у освоєнні, але водночас надавати достатньо інформації про результати аналізу.

Для реалізації клієнтської частини було обрано підхід на основі HTML, CSS та JavaScript з використанням фреймворку Bootstrap 5. Це дозволило швидко створити адаптивний дизайн та використати готові компоненти для основних елементів керування та відображення. Застосунок реалізовано як односторінковий (Single Page Application – SPA), де перехід між основними функціональними розділами відбувається динамічно, без перезавантаження всієї вебсторінки, що забезпечує кращий користувацький досвід.

Навігація між розділами (список розділів відображено на рисунку 2.6) реалізована за допомогою верхньої навігаційної панелі (хедер), створеної з використанням компонента Navbar з Bootstrap. Кожне посилання в панелі активує відображення відповідної секції на сторінці та приховує інші.



Рисунок 2.6 – Структура сторінок

При проєктуванні користувацького інтерфейсу вебзастосунку головний акцент робився на створенні інтуїтивно зрозумілого та ефективного середовища для взаємодії. Керуючись ключовими принципами UI/UX, вдалося забезпечити позитивний досвід для користувачів з різним рівнем технічної підготовки. Ключові елементи UI та їх UX-призначення наведено в таблиці 2.3.

Таблиця 2.3 – Ключові елементи UI та їх UX-призначення

Елемент UI	**UX-Призначення**	**Технологія / Компонент**
Навігаційна панель	Швидкий доступ до основних розділів застосунку.	Bootstrap Navbar
Форма завантаження файлу	Простий та зрозумілий спосіб надання аудіо для аналізу.	HTML <input type="file">, Bootstrap форми
Кнопка «Розпізнати емоцію»	Чіткий заклик до дії для початку аналізу.	Bootstrap Button
Індикатор завантаження	Інформування користувача про те, що система обробляє запит.	Bootstrap Spinner
Відображення передбаченої емоції/впевненості	Швидке отримання основного результату аналізу.	Динамічний HTML (JS)

Продовження таблиці 2.3

Елемент UI	**UX-Призначення**	**Технологія / Компонент**
Діаграма ймовірностей емоцій	Наочне представлення розподілу ймовірностей, альтернативні варіанти.	Chart.js (стовпчикова)
Графік зміни емоцій у часі (для довгих файлів)	Візуалізація динаміки емоцій протягом тривалого аудіозапису.	Chart.js (лінійний)
Текстовий опис емоції	Додаткова інтерпретація результату, що робить його більш зрозумілим.	Динамічний HTML (JS)
Список історії аналізів	Можливість перегляду попередніх результатів, відстеження.	Bootstrap List group
Галерея прикладів спектрограм	Навчальний елемент, допомагає користувачеві краще розуміти спектрограми.	Bootstrap Card

Насамперед, було дотримано принципу простоти та мінімалізму. Інтерфейс свідомо не перевантажувався зайвими елементами, а основний фокус було зосереджено на ключовій функціональності – аналізі емоцій. Це дозволило уникнути візуального шуму та полегшити користувачам навігацію й розуміння доступних дій. Важливим аспектом став зворотний зв’язок із системою: користувач завжди отримує актуальну інформацію про поточний стан процесу, чи то завантаження файлу, його обробка на сервері, чи виникнення можливих помилок. Це реалізовано через візуальні індикатори та інформативні повідомлення.

Завдяки використанню фреймворку Bootstrap, було забезпечено адаптивність дизайну. Інтерфейс коректно масштабується та перебудовується для комфортного відображення на пристроях з різними розмірами екранів, від настільних комп’ютерів до планшетів та смартфонів. Візуальна привабливість досягалася шляхом застосування стандартних, але акуратних та сучасних компонентів Bootstrap, які були доповнені кастомними стилями для створення

унікального вигляду. Крім того, наочна візуалізація результатів аналізу за допомогою графіків та спектрограм значно покращує сприйняття інформації. Зрештою, прагнення до інтуїтивності було наскрізним: весь процес аналізу, від завантаження файлу до інтерпретації результатів, спроєктовано таким чином, щоб він був максимально зрозумілим навіть для користувача, який не має спеціальних знань у галузі обробки аудіо чи машинного навчання.

2.5 Висновки до розділу

У другому розділі кваліфікаційної роботи було детально розглянуто процес проєктування та розробки вебзастосунку для автоматичного аналізу емоцій у голосі. Вибір технологічного стеку був ретельно обґрунтований: для серверної частини та розробки моделі машинного навчання було обрано Python з фреймворком FastAPI та бібліотеками PyTorch, Librosa, Scikit-learn, що забезпечило необхідну потужність для обробки аудіо та реалізації глибокої нейронної мережі. Клієнтська частина була реалізована з використанням HTML, CSS, JavaScript та фреймворку Bootstrap 5, що дозволило створити адаптивний та функціональний користувацький інтерфейс з мінімальними зусиллями на складну фронтенд-логіку.

Архітектура системи побудована за класичною клієнт-серверною моделлю з чітко визначеним REST API для взаємодії. Було спроєктовано ключові ендпоінти для аналізу аудіо (/api/audio-analyze) та отримання історії аналізів (/api/history), детально описано формати запитів та відповідей, включаючи обробку як коротких, так і довгих аудіофайлів із застосуванням сегментації для останніх.

3 РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ВЕБЗАСТОСУНКУ

3.1 Реалізація серверної частини (Backend) на FastAPI

Серверна частина вебзастосунку, відповідальна за обробку запитів від клієнта, взаємодію з моделлю машинного навчання та управління даними, була реалізована на мові Python з використанням вебфреймворку FastAPI [11].

Основним ендпоінтом серверної частини є /api/audio-analyze, який обробляє POST-запити з аудіофайлами для аналізу. При зверненні до нього FastAPI автоматично приймає завантажений файл і виконує базову перевірку його наявності. Далі аудіо конвертується у стандартизований формат: WAV, моно, з частотою дискретизації 22050 Гц, що забезпечує єдиний вхідний формат для обробки – ця трансформація виконується за допомогою бібліотеки Pydub [15].

Після цього визначається тривалість аудіозапису. Якщо вона перевищує встановлений поріг, файл розбивається на менші сегменти із заданою довжиною та перекриттям. У випадку короткого аудіо сегментація не проводиться. Кожен сегмент, або весь файл, якщо він невеликий, експортується у тимчасовий .wav-файл, з якого аудіосигнал зчитується за допомогою librosa.load.

Далі формується мел-спектрограма – спочатку через librosa.feature.melspectrogram, а потім вона перетворюється у шкалу децибелів. Щоб відповідати вимогам моделі, спектрограма приводиться до фіксованого розміру, після чого нормалізується: віднімається середнє значення та ділиться на стандартне відхилення, попередньо обчислені на тренувальному наборі. Підготовлена у такий спосіб спектрограма передається до моделі, яка виконує передбачення емоційного стану. Навчена модель SpectrogramCNN, об’єкт LabelEncoder та параметри нормалізації завантажуються при старті FastAPI застосунку.

Завантаження моделі відбувається за стандартною процедурою: спочатку створюється екземпляр класу SpectrogramCNN з тими самими параметрами архітектури, що використовувалися під час навчання. Потім за допомогою torch.load() завантажується словник стану з файла .pth, який містить збережені ваги та зміщення, як показано в коді на рисунку 3.1. Ці параметри застосовуються до моделі через метод load_state_dict(), після чого модель переводиться в режим оцінки за допомогою model.eval(), що забезпечує коректну роботу шарів, таких як Dropout та BatchNorm, під час інференсу.Наступним виконується завантаження об’єктів з файлів .pkl за допомогою pickle.load(). Ці об’єкти використовуються для перетворення передбачених індексів класів у текстові мітки емоцій та для нормалізації вхідних спектрограм відповідно.

```
# main.py
MODEL_PATH_API = os.path.join(MODEL_DIR_API, 'model_cnn2d_spectrogram.pth')
ENCODER_PATH_API = os.path.join(MODEL_DIR_API, 'label_encoder_spectrogram.pkl')
SCALER_PATH_API = os.path.join(MODEL_DIR_API, 'spectrogram_scaler.pkl')
try:
    # Завантаження LabelEncoder
    with open(ENCODER_PATH_API, 'rb') as f:
        label_encoder = pickle.load(f)
    NUM_CLASSES_API = len(label_encoder.classes_)

    # Завантаження параметрів нормалізації
    with open(SCALER_PATH_API, 'rb') as f:
        scaler_params = pickle.load(f)
    DATASET_MEAN = scaler_params['mean']
    DATASET_STD = scaler_params['std']
    # Ініціалізація та завантаження моделі
    model_instance = SpectrogramCNN(N_MELS, FIXED_TIME_STEPS,
NUM_CLASSES_API, DROPOUT_RATE)
    model_instance.load_state_dict(torch.load(MODEL_PATH_API,
map_location=torch.device('cpu')))
    model_instance.eval() # Перевести модель в режим оцінки
    model = model_instance # Глобальна змінна для використання в ендпоінтах
    print(f"CNN2D Spectrogram model loaded successfully.")
except Exception as e:
    print(f"Error loading model or related files: {e}")
    exit()
```

Рисунок 3.1 – Фрагмент коду завантаження моделі та артефактів

Під час передбачення, підготовлений тензор спектрограми передається методу forward завантаженої моделі. Отримані логіти пропускаються через функцію F.softmax для отримання ймовірностей класів, після чого обирається клас з максимальною ймовірністю.

3.2 Реалізація клієнтської частини

Клієнтська частина реалізована як односторінковий застосунок (SPA), де навігація між розділами відбувається динамічно за допомогою JavaScript. Основу складає файл index.html, стилі визначені у css/style.css, а вся логіка взаємодії – у js/script.js.

Використовується стандартна HTML-форма. Обробник події submit для цієї форми в js/script.js виконує наступні дії:

– запобігає стандартній відправці форми;
– отримує вибраний файл з <input type="file">;
– створює об’єкт FormData та додає до нього файл;
– відображає індикатор завантаження;
– за допомогою axios.post відправляє FormData на ендпоінт /api/audio-analyze;
– обробляє відповідь від сервера або помилку.

Після отримання успішної відповіді від ендпоінту /api/audio-analyze, JavaScript динамічно генерує HTML-код для відображення результатів у спеціально відведеному блоці, основну ідею коду показано на рисунку 3.2.

Для коротких файлів користувач отримує такий результат:

– відображається назва передбаченої емоції та її впевненість;
– поруч виводяться два зображення: спектрограма аналізованого аудіо (отримана як base64 рядок від API) та приклад спектрограми для передбаченої емоції (шлях до статичного зображення) (зображення клікабельні для відкриття у модальному вікні Bootstrap);

– за допомогою Chart.js будується стовпчикова діаграма розподілу ймовірностей для всіх класів емоцій;
– виводиться короткий текстовий опис, що характеризує передбачену емоцію.

Для довгих файлів:

– відображається повідомлення про те, що файл був оброблений по сегментах;
– виводиться домінантна емоція для всього файлу;
– за допомогою Chart.js будується лінійний графік, що показує зміну ймовірностей основних емоцій по часових сегментах;
– відображається спектрограма першого сегмента файлу та приклад спектрограми для домінантної емоції;
– детальна інформація по кожному сегменту (час, емоція, впевненість, ймовірності) виводиться у вигляді списку (Bootstrap Accordion).

```
// js/script.js (всередині обробника відповіді від axios.post)
if (response.data && resultArea) {
    let htmlResult = `<h2 class="text-center mb-4">Результати аналізу:</h2>`;
    if (response.data.is_long_audio) {
        // Логіка для довгого аудіо
        htmlResult += `<!-- ... HTML для переважаючої емоції, графіка зміни емоцій, спектрограм, акордеону ... -->`;
        resultArea.innerHTML = htmlResult;
        if (response.data.segment_analysis && document.getElementById('emotionTimelineChart')) {
            renderEmotionTimelineChart(response.data.segment_analysis); // Функція для малювання графіка
        }
    } else {
        // Логіка для короткого аудіо
        htmlResult += `<!-- ... HTML для передбаченої емоції, спектрограм, діаграми ймовірностей, опису ... -->`;
        resultArea.innerHTML = htmlResult;
        if (response.data.probabilities && document.getElementById('probabilitiesChart')
```

Рисунок 3.2 – Фрагмент коду візуалізації результату в js/script.js

Сторінка «Історія аналізів» при активації завантажує дані з ендпоінту /api/history за допомогою axios.get. Отриманий масив записів динамічно

перетворюється на список елементів (Bootstrap list-group), де кожен елемент відображає ключову інформацію про аналіз та мініатюри спектрограм (для коротких файлів).

Сторінка «Приклади» містить статичну HTML-розмітку з картками (Bootstrap card), що відображають попередньо згенеровані зображення спектрограм для кожної з цільових емоцій.

3.3 Процес навчання, валідації та тестування моделі машинного навчання

Ключовим етапом розробки системи стало навчання та оцінка моделі SpectrogramCNN, призначеної для класифікації емоцій за мел-спектрограмами. Цей процес включав ретельну підготовку даних, визначення архітектури, вибір параметрів навчання та всебічну оцінку отриманих результатів.

Для навчання моделі було використано публічний датасет RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song). З нього було відібрано аудіозаписи, що відповідають п’яти цільовим емоційним станам: спокій, радість, сум, злість та страх. Кожен аудіофайл був перетворений на мел-спектрограму з наступними параметрами: частота дискретизації SAMPLE_RATE = 22050 Гц, розмір вікна FFT N_FFT = 2048, крок вікна HOP_LENGTH = 512, кількість мел-смуг N_MELS = 128. Для забезпечення однакового розміру вхідних даних для нейронної мережі, всі спектрограми приводилися до фіксованої ширини FIXED_TIME_STEPS = 256 шляхом падингу або обрізання. Отримані спектрограми логарифмувалися (librosa.power_to_db) та нормалізувалися шляхом віднімання середнього значення та ділення на стандартне відхилення, розрахованих виключно на тренувальній вибірці.

Загальний набір даних було розділено на три частини: тренувальну (~70%), валідаційну (~15%) та тестову (~15%) для об’єктивної оцінки моделі.

Емоційні мітки були перетворені на числові значення за допомогою sklearn.preprocessing.LabelEncoder.

Для збільшення обсягу тренувальних даних та покращення узагальнюючої здатності моделі застосовувалася аугментація спектрограм. Цей процес включав випадкове маскування часових смуг (Time Masking) та частотних смуг (Frequency Masking) на спектрограмах тренувальної вибірки. Параметри маскування підбиралися таким чином, щоб внести достатню варіативність, не спотворюючи при цьому ключові емоційні ознаки.

Навчання моделі SpectrogramCNN, проводилося з використанням фреймворку PyTorch. Для подачі даних моделі батчами використовувалися об’єкти torch.utils.data.DataLoader з параметром BATCH_SIZE = 16 та увімкненим перемішуванням для тренувальної вибірки. Ключові параметри та компоненти процесу навчання наведено в таблиці 3.1.

Таблиця 3.1 – Параметри навчання моделі SpectrogramCNN

Параметр / Компонент	**Значення / Опис**
Функція втрат (Loss Function)	nn.CrossEntropyLoss
Оптимізатор (Optimizer)	optim.AdamW (Adam з корекцією L2-регуляризації)
Початкова швидкість навчання	LEARNING_RATE = 0.0005
L2-регуляризація (Weight Decay)	1e-4 (застосовано в оптимізаторі AdamW)
Планувальник швидкості навчання	optim.lr_scheduler.ReduceLROnPlateau (зменшення LR при плато валідаційної втрати)
Фактор зменшення LR	0.2
«Терпіння» планувальника	5 епох без покращення val_loss
Максимальна кількість епох	EPOCHS = 150
Рання зупинка (Early Stopping)	Припинення навчання, якщо val_loss не покращується протягом PATIENCE = 15 епох
Пристрій для навчання	CPU (з можливістю перемикання на CUDA, якщо доступно)

Протягом кожної епохи модель спочатку переводилася в режим навчання (model.train()). На цьому етапі обчислювалися градієнти та

оновлювалися ваги моделі на основі тренувальних даних. Потім модель переводилася в режим оцінки (model.eval()) для розрахунку втрат та точності на валідаційній вибірці без оновлення ваги. Цей процес дозволяв відстежувати ефективність навчання та запобігати перенавчанню. Модель з найкращим (найменшим) значенням валідаційної втрати зберігалася для подальшого використання, метод, яким це було реалізовано знаходиться на рисунку 3.3.

```
# train_cnn2d_spectrogram.py
# ... (ініціалізація моделі, criterion, optimizer, scheduler) ...
for epoch in range(EPOCHS):
    model.train()
    # ... (цикл по train_loader, обчислення train_loss, train_acc) ...
    model.eval()
    with torch.no_grad():
        # ... (цикл по val_loader, обчислення val_loss, val_acc) ...
    scheduler.step(epoch_val_loss) # Оновлення швидкості навчання
    if epoch_val_loss < best_val_loss:
        best_val_loss = epoch_val_loss
        torch.save(model.state_dict(), MODEL_PATH) # Збереження найкращої моделі
        epochs_no_improve = 0
    else:
        epochs_no_improve += 1
        if epochs_no_improve == EARLY_STOPPING_PATIENCE:
            print("Early stopping triggered.")
            break
# ...
```

Рисунок 3.3 – Фрагмент коду навчального циклу

Після завершення навчання, найкраща збережена модель за показником валідаційної втрати була оцінена на відкладеній тестовій вибірці. Це забезпечило об’єктивну оцінку її здатності узагальнювати на нових, не бачених раніше даних. Для оцінки використовувалися метрики: загальна точність (accuracy), точність по класах (precision), повнота (recall), F1-міра (F1-score) та матриця помилок (confusion matrix).

Отримані показники дозволили зробити висновки про ефективність розробленої моделі та визначити її сильні та слабкі сторони у розпізнаванні різних емоційних станів. Графіки кривих навчання (залежність втрат та точності від епохи) також були збережені для аналізу динаміки процесу навчання.

3.4 Тестування функціональності вебзастосунку

Паралельно з оцінкою моделі проводилося тестування функціональності розробленого вебзастосунку. Основною метою цього етапу було переконатися, що всі компоненти системи взаємодіють коректно, а користувацький інтерфейс є зручним та відповідає заявленим вимогам.

Тестування проводилося вручну за заздалегідь розробленими сценаріями, що охоплювали основні варіанти використання застосунку:

- аналіз короткого аудіофайлу: завантаження файлу тривалістю до 12 секунд, перевірка коректності відображення передбаченої емоції, впевненості, спектрограм (аналізованої та прикладу), діаграми ймовірностей та текстового опису;
- аналіз довгого аудіофайлу: завантаження файлу тривалістю понад 12 секунд, перевірка відображення повідомлення про сегментарний аналіз, переважаючої емоції, графіка зміни емоцій у часі, спектрограми першого сегмента та прикладу, а також деталізації по сегментах в акордеоні;
- обробка невалідних вхідних даних: спроби завантаження файлів не у форматі .wav або пошкоджених файлів для перевірки механізмів обробки помилок та відповідних повідомлень користувачеві;
- робота зі сторінкою «Історія аналізів»: перевірка завантаження та коректного відображення списку попередніх аналізів, включаючи всю релевантну інформацію та функціональність модальних вікон для спектрограм (перевірка локального очищення історії);
- робота зі сторінкою «Приклади»: перевірка відображення галереї прикладів спектрограм;
- загальна навігація та адаптивність: тестування переходів між сторінками та коректності відображення інтерфейсу на різних розмірах екранів (використовуючи інструменти розробника браузера для емуляції).

Під час тестування основний функціонал застосунку демонстрував стабільну роботу. Було підтверджено, що клієнтська частина коректно взаємодіє з API, а результати аналізу представляються користувачеві у зрозумілому та інформативному вигляді. Виявлені дрібні недоліки в інтерфейсі або логіці відображення виправлялися ітераційно. Загалом, вебзастосунок було визнано готовим до демонстрації та відповідаючим поставленим у проєкті завданням.

Для більш детального та ізольованого тестування серверної частини (API) було використано інструмент Postman. Це популярна платформа для розробки та тестування API, що дозволяє створювати та відправляти HTTP-запити, переглядати відповіді від сервера, а також автоматизувати тестування за допомогою колекцій та скриптів. Використання Postman дозволило перевірити коректність роботи ендпоінтів незалежно від клієнтської частини, валідувати структуру запитів та відповідей, а також оцінити поведінку API при різних вхідних даних.

Було створено колекцію запитів у Postman (див. рис. 3.4 та 3.5), що охоплювала всі основні ендпоінти розробленого API.

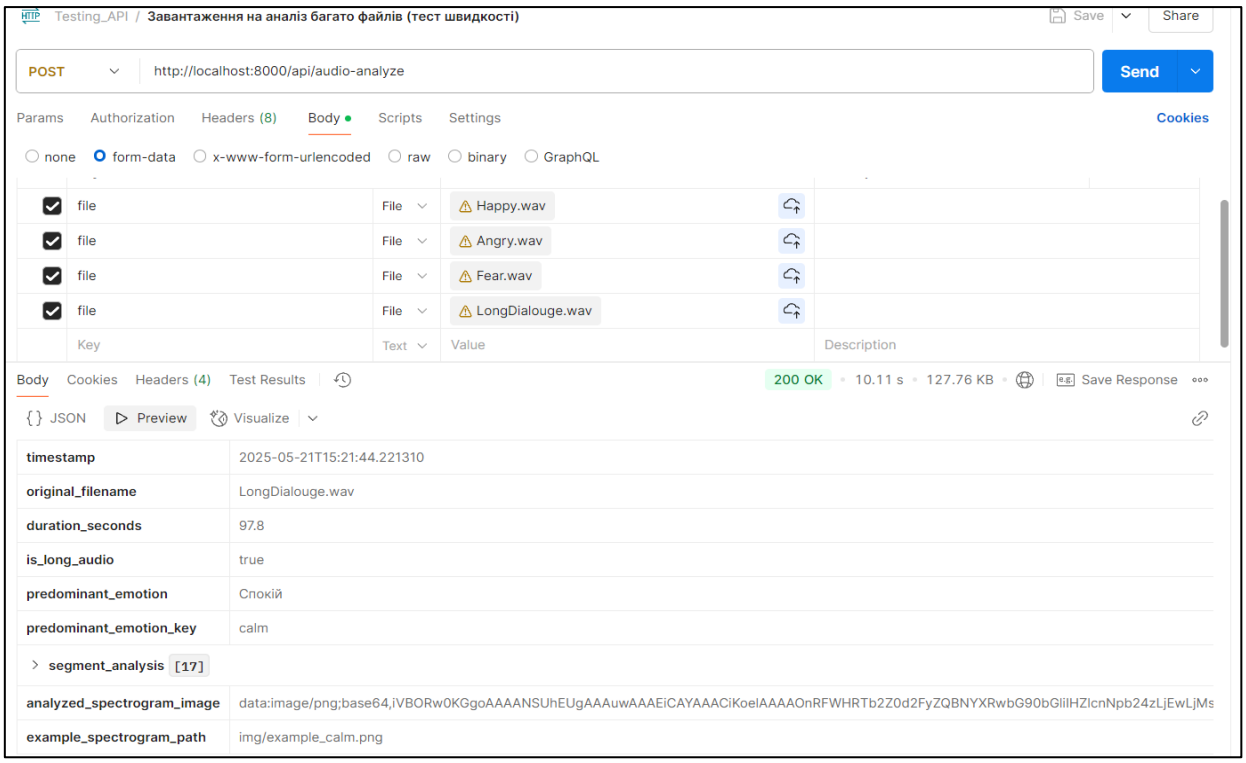


Рисунок 3.4 – Тестування ендпоінту POST /api/audio-analyze

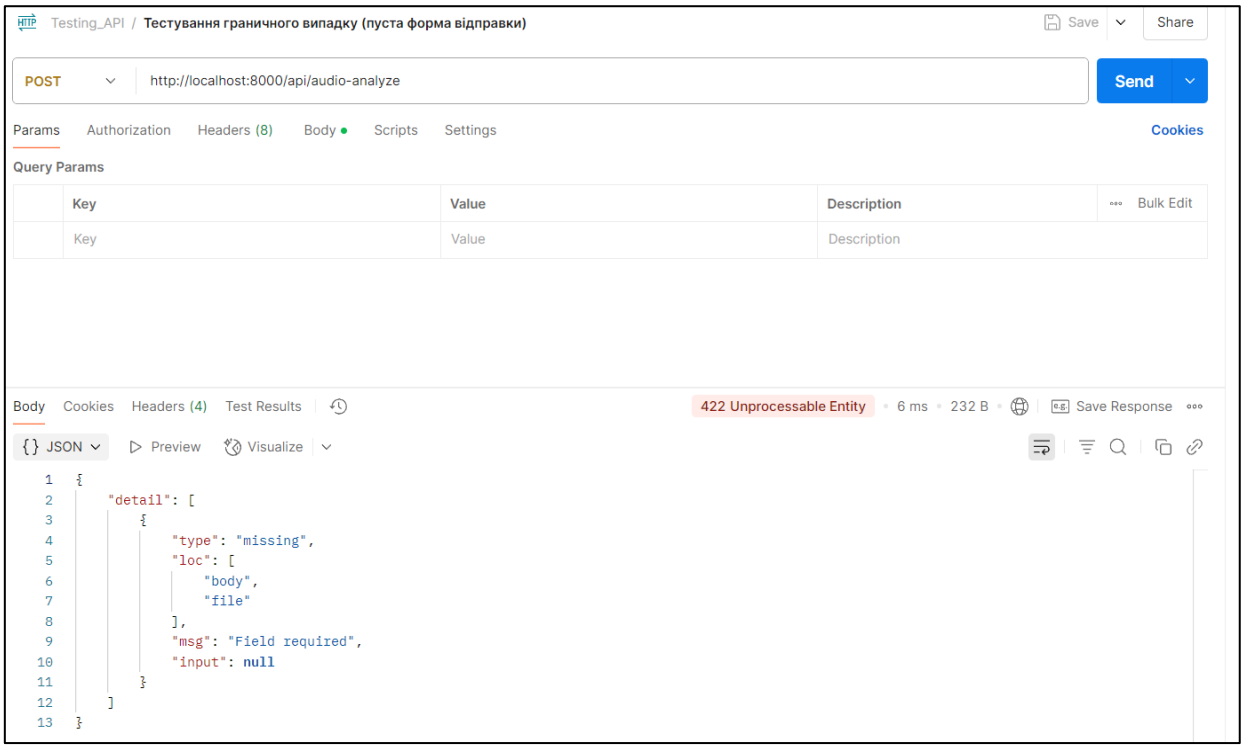


Рисунок 3.5 – Тестування POST /api/audio-analyze (граничний випадок)

Для кожного запиту визначалися HTTP-метод, URL, необхідні заголовки та тіло запиту.

3.5 Аналіз результатів роботи системи розпізнавання емоцій та обговорення перспектив її розвитку

Розроблений вебзастосунок та інтегрована система розпізнавання емоцій у голосі продемонстрували свою працездатність та здатність вирішувати поставлені завдання. Практична реалізація, що охоплювала розробку моделі машинного навчання, серверної API та клієнтського інтерфейсу, дозволила отримати функціональний прототип, результати роботи якого підлягають подальшому аналізу.

Оцінка якості моделі SpectrogramCNN на відкладеній тестовій вибірці датасету RAVDESS показала загальну точність класифікації на рівні 74% для п’яти цільових емоцій: спокій, радість, сум, злість, страх. Цей показник є досить високим, враховуючи складність задачі та використання кастомної

архітектури CNN, навченої з нуля. Аналіз звіту про класифікацію та матриці помилок виявив, що модель найкраще розпізнає емоції «злість» та «спокій» (F1-міра 0.86 та 0.82 відповідно), демонструючи хороші показники для «страху» (F1-міра 0.75) та задовільні для «суму» (F1-міра 0.69). Найбільші труднощі виникли з розпізнаванням емоції «радість» (F1-міра 0.55), яку модель частіше плутала з іншими емоціями. Це узгоджується з результатами багатьох досліджень у цій галузі, де вокальні прояви радості можуть бути дуже варіативними та перетинатися з ознаками інших емоційних станів. Криві навчання показали, що застосований механізм ранньої зупинки ефективно запобіг глибокому перенавчанню моделі, зупинивши процес на етапі, коли валідаційна втрата досягла свого мінімуму.

Реалізована серверна частина на FastAPI забезпечила стабільну роботу API, коректну обробку аудіофайлів, включаючи їх сегментацію для аналізу довгих записів, та взаємодію з моделлю машинного навчання. Тестування API за допомогою інструменту Postman підтвердило відповідність ендпоінтів проєктним специфікаціям та їх здатність обробляти різні типи запитів, включаючи граничні та некоректні випадки. Збереження історії аналізів у JSON-файл є простим, але робочим рішенням для прототипу, хоча для промислового використання потребувало б переходу на більш надійні системи управління базами даних.

Клієнтська частина вебзастосунку, розроблена з використанням HTML, CSS, JavaScript та Bootstrap, надала користувачам інтуїтивно зрозумілий інтерфейс для взаємодії з системою. Візуалізація результатів аналізу, зокрема відображення передбаченої емоції, її впевненості, спектрограм (аналізованої та прикладу), а також графіків розподілу ймовірностей (для коротких файлів) та динаміки емоцій у часі (для довгих файлів), суттєво підвищила інформативність та наочність результатів. Можливість перегляду історії аналізів та прикладів спектрограм для різних емоцій також сприяє кращому розумінню роботи системи та інтерпретації її передбачень. Тестування користувацького інтерфейсу підтвердило його функціональність та адаптивність.

Попри досягнуті результати, система має потенціал для подальшого вдосконалення. Насамперед, це стосується підвищення точності моделі. Застосування більш розвинених методів аугментації, таких як повна реалізація SpecAugment, додавання фонового шуму, варіації швидкості та висоти тону, може значно покращити її здатність до узагальнення. Також перспективним напрямком є використання складніших архітектур – наприклад, CRNN або трансформерів на зразок Wav2Vec2 чи HuBERT. Додаткове навчання на різноманітних корпусах емоційного мовлення, з урахуванням мовних і культурних особливостей, дозволить розширити охоплення моделі. Окрему увагу варто приділити й оптимізації гіперпараметрів за допомогою сучасних підходів, таких як Grid Search або Bayesian Optimization.

Ще одним важливим напрямком є покращення обробки довгих аудіофайлів. Замість простої агрегації результатів, доцільно реалізувати виявлення ключових емоційних епізодів, оцінку загального емоційного фону або відстеження переходів між емоціями. Удосконалення візуалізації змін емоцій у часі – наприклад, шляхом додавання інтерактивності – також посилить користувацький досвід.

У плані розширення функціоналу перспективним є впровадження «живого» аналізу з мікрофона в режимі, наближеному до реального часу. Крім того, важливо забезпечити зручне керування історією аналізів, зокрема можливість видаляти окремі записи. Це може потребувати переходу на використання бази даних та реалізації облікових записів користувачів для персоналізації доступу. Інтеграція зворотного зв’язку від користувача щодо точності передбачень також відкриває можливості для подальшого донавчання моделі.

Загалом створена система становить надійну базу для подальших досліджень і розвитку в галузі автоматичного аналізу емоцій у голосі. Отримані результати підтверджують життєздатність обраного підходу, а окреслені напрямки дозволяють рухатися до створення ще точнішого, функціональнішого та кориснішого інструменту.

3.6 Висновки до розділу

У третьому розділі було представлено практичну реалізацію вебзастосунку для аналізу емоцій у голосі та процес навчання й тестування його ключового компонента – моделі машинного навчання. Реалізація серверної частини на FastAPI продемонструвала ефективність обраного фреймворку для створення API, обробки аудіофайлів (включаючи сегментацію довгих записів) та інтеграції з навченою PyTorch моделлю SpectrogramCNN. Клієнтська частина, побудована на HTML, CSS, JavaScript з Bootstrap, забезпечила інтерактивний та адаптивний користувацький інтерфейс для завантаження файлів, візуалізації результатів аналізу (включаючи спектрограми та графіки ймовірностей/динаміки емоцій) та перегляду історії.

Детально описано процес навчання моделі SpectrogramCNN на датасеті RAVDESS, включаючи етапи підготовки та аугментації даних (мел-спектрограм), вибір параметрів навчального циклу, використання механізмів планування швидкості навчання та ранньої зупинки. Наголошено на важливості оцінки моделі на відкладеній тестовій вибірці для отримання об’єктивних показників її ефективності.

Також було розглянуто підхід до тестування функціональності всього вебзастосунку за допомогою набору сценаріїв, що охоплюють основні варіанти взаємодії користувача з системою. Це дозволило перевірити коректність роботи всіх реалізованих функцій та зручність користувацького інтерфейсу. Таким чином, даний розділ підсумовує етап втілення проєктних рішень у працюючий програмний продукт, результати детального аналізу ефективності якого будуть представлені у наступному розділі.

ВИСНОВКИ

Кваліфікаційну роботу присвячено вирішенню актуальної науково-технічної задачі автоматичного аналізу емоцій у людському голосі за допомогою методів машинного навчання. Для досягнення мети було проведено комплексне дослідження предметної області, спроєктовано архітектуру системи, розроблено та виконано навчання відповідної моделі, реалізовано серверну та клієнтську частини вебзастосунку, а також проведено його тестування.

У першому розділі представлено аналіз теоретичних основ розпізнавання емоцій у мовленні. Розглянуто психофізіологічні аспекти прояву емоцій у голосі, проаналізовано ключові акустичні ознаки, такі як просодичні, спектральні та кепстральні характеристики, що несуть інформацію про емоційний стан. Особливу увагу приділено мел-спектрограмам як ефективному представленню аудіосигналу для подальшого аналізу за допомогою глибоких нейронних мереж. Також було досліджено існуючі підходи до машинного навчання в задачах класифікації емоцій, від класичних алгоритмів до сучасних згорткових та рекурентних архітектур, та проаналізовано характеристики поширених датасетів, зокрема RAVDESS, який було обрано основним для навчання моделі.

Проєктування системи базувалося на клієнт-серверній архітектурі. Для серверної частини було обрано мову програмування Python та фреймворк FastAPI, що дозволило створити високопродуктивний REST API. Модель машинного навчання, SpectrogramCNN, була розроблена та навчена з використанням бібліотеки PyTorch. Вона приймає на вхід мел-спектрограми та класифікує їх на п’ять емоційних станів: спокій, радість, сум, злість та страх. Клієнтська частина реалізована на HTML, CSS та JavaScript з використанням фреймворку Bootstrap 5, що забезпечило створення адаптивного та інтуїтивно зрозумілого користувацького інтерфейсу. Взаємодія між клієнтом та сервером

здійснюється через чітко визначені API ендпоінти, що обробляють завантаження аудіофайлів, аналіз емоцій та надання історії попередніх аналізів. Було передбачено обробку як коротких, так і довгих аудіозаписів; для останніх реалізовано механізм сегментації та визначення переважаючої емоції разом з візуалізацією динаміки емоцій у часі.

Практична реалізація включала написання програмного коду для всіх компонентів системи. Модель SpectrogramCNN була навчена на підготовлених даних з датасету RAVDESS, із застосуванням технік аугментації для покращення узагальнення. Процес навчання контролювався за допомогою валідаційної вибірки та механізмів планування швидкості навчання й ранньої зупинки. Тестування навченої моделі на відкладеній тестовій вибірці продемонструвало загальну точність на рівні 74%, що є хорошим результатом для даної задачі та обраного підходу. Найкраще модель розпізнавала емоції «злість» та «спокій», тоді як найбільші труднощі виникли з емоцією «радість». Функціональне тестування вебзастосунку, включаючи тестування API за допомогою Postman та ручне тестування користувацького інтерфейсу, підтвердило коректність роботи всіх основних функцій та зручність взаємодії з системою.

Таким чином, у ході виконання кваліфікаційної роботи було досягнуто поставленої мети: розроблено та реалізовано вебзастосунок, що дозволяє автоматично аналізувати емоції у голосі. Створений програмний продукт має практичну цінність і може бути використаний як інструмент для подальших досліджень у галузі розпізнавання емоцій, в освітніх цілях для демонстрації можливостей машинного навчання, а також як основа для розробки більш складних систем моніторингу емоційного стану в різних сферах застосування.

Отримані результати підтверджують ефективність використання згорткових нейронних мереж для аналізу мел-спектрограм у задачах розпізнавання емоцій. Водночас, проведений аналіз виявив потенційні напрямки для подальшого вдосконалення системи, такі як застосування більш просунутих технік аугментації даних, експерименти зі складнішими

архітектурами моделей (наприклад, CRNN або трансформери), розширення функціоналу вебзастосунку (зокрема, повноцінна реалізація live-аналізу та розширені можливості управління історією), а також оптимізація продуктивності шляхом використання GPU для інференсу на сервері. Подальші дослідження також можуть бути спрямовані на адаптацію моделі для роботи з іншими мовами, включаючи українську, та врахування культурних особливостей вираження емоцій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Schiewer G. L., Altarriba J., Ng B. C. Language and Emotion: An International Handbook. Berlin : De Gruyter Mouton, 2022. 850 p.

2. Corkett-Beirne S. Language and Emotion: A Study Investigating Interjectional Usage by Children and Adolescents with Autism Spectrum Disorder, Developmental Language Disorder, and a Typically Developing Cohort. Hamburg : Bedey Media GmbH, 2022. 120 p.

3. McLaren K. The Language of Emotions: What Your Feelings Are Trying to Tell You (Revised Edition). Boulder, CO : Sounds True, 2023. 320 p.

4. Koolagudi S. G., Rao K. S. Speech Emotion Recognition: State of the Art. Cham : Springer, 2021. 180 p.

5. Ayadi M. E., Kamel M. S., Karray F. Survey on Speech Emotion Recognition: Features, Classification Schemes, and Databases. *Emotions and Affect in Human Factors and Human-Computer Interaction* : 2nd ed. Amsterdam : Elsevier, 2021. P. 285-312.

6. Zhang S., Schuller B. W., Cummins N. Recent Advances and Challenges in Speech Emotion Recognition with Deep Learning. *IEEE Transactions on Affective Computing*. 2022. Vol. 13, Issue 2. P. 803-820.

7. Sahidullah M., Saha G., Kinnunen T. Fundamentals of Speaker Recognition. Cham : Springer, 2020. 250 p.

8. Rabiner L., Schafer R. Theory and Applications of Digital Speech Processing. Upper Saddle River, NJ : Pearson, 2011. 768 p.

9. Eyben F., Scherer K. R., Schuller B. W. et al. The Geneva Minimalistic Acoustic Parameter Set (GeMAPS) for Voice Research and Affective Computing. *IEEE Transactions on Affective Computing*. 2016. Vol. 7, Issue 2. P. 190-202.

10. Python documentation. *Python.org*. URL: https://docs.python.org/3/ (дата звернення: 13.02.2025).

11. FastAPI. *Tiangolo*. URL: https://fastapi.tiangolo.com/ (дата звернення: 15.02.2025).

12. PyTorch Documentation. *PyTorch*. URL: https://pytorch.org/docs/stable/index.html (дата звернення: 15.02.2025).

13. Scikit-learn Documentation. *Scikit-learn*. URL: https://scikit-learn.org/stable/documentation.html (дата звернення: 19.02.2025).

14. Librosa documentation. *Librosa*. URL: https://librosa.org/doc/latest/index.html (дата звернення: 19.02.2025).

15. Pydub. *Pydub*. URL: https://pydub.com/ (дата звернення: 19.03.2025).

16. Matplotlib documentation. *Matplotlib*. URL: https://matplotlib.org/stable/contents.html (дата звернення: 19.03.2025).

17. Introduction. *Bootstrap 5*. URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 07.03.2025).

18. Introduction. *Axios*. URL: https://axios-http.com/docs/intro (дата звернення: 08.03.2025).

19. Documentation. *Chart.js*. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 08.03.2025).

20. MediaRecorder API. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/MediaRecorder_API (дата звернення: 15.03.2025).

21. RAVDESS dataset. *Zenodo*. URL: https://zenodo.org/record/1188976 (дата звернення: 24.03.2025).

22. EMO-DB dataset. *Institute of Phonetics and Speech Processing, LMU Munich*. URL: https://www.phonetik.uni-muenchen.de/Bas/BasEmo.html (дата звернення: 25.03.2025).

23. IEMOCAP. *SAIL* (Signal Analysis and Interpretation Laboratory), USC. URL: https://sail.usc.edu/iemocap/ (дата звернення: 25.03.2025).

24. SAVEE dataset. University of Surrey. URL: http://kahlan.eps.surrey.ac.uk/savee/ (дата звернення: 25.03.2025).

ДОДАТОК А

Програмна реалізація API

```
import os
import io
import pickle
import tempfile
import numpy as np
import librosa
import librosa.display # Потрібен для specshow
import torch
import torch.nn as nn
import torch.nn.functional as F
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse
from fastapi.middleware.cors import CORSMiddleware
from pydub import AudioSegment
from pydub.utils import make_chunks # Для нарізки аудіо
import matplotlib.pyplot as plt
import base64
import json
from datetime import datetime
from collections import Counter # Для підрахунку найчастішої емоції

# --- Налаштування ---
SAMPLE_RATE = 22050      # Цільова частота дискретизації для аудіо
N_FFT = 2048             # Розмір вікна для Швидкого Перетворення Фур'є
HOP_LENGTH = 512         # Крок вікна FFT, впливає на ширину спектрограми
N_MELS = 128             # Кількість мел-фільтрів, висота спектрограми
FIXED_TIME_STEPS = 256   # Фіксована кількість часових кроків для спектрограми
DROPOUT_RATE = 0.4       # Коефіцієнт Dropout для моделі

# Словник для перекладу англійських ключів емоцій на українські назви
EMOTION_TRANSLATION_MAP = {
    'angry': 'Злість', 'calm': 'Спокій', 'fearful': 'Страх',
    'happy': 'Радість', 'sad': 'Сум',
}

# Шляхи до зображень-прикладів спектрограм (відносні, для фронтенду)
EXAMPLE_SPECTROGRAM_PATHS = {
    'calm': 'img/example_calm.png',
```

```
    'happy': 'img/example_happy.png',
    'sad': 'img/example_sad.png',
    'angry': 'img/example_angry.png',
    'fearful': 'img/example_fearful.png',
}
HISTORY_FILE_PATH = "analysis_history.json" # Файл для зберігання історії аналізів

# --- Параметри для сегментації довгих аудіофайлів ---
LONG_AUDIO_THRESHOLD_SECONDS = 12.0  # Поріг, після якого аудіо вважається "довгим"
# Тривалість одного сегмента, розрахована на основі параметрів моделі
SEGMENT_DURATION_SECONDS = (FIXED_TIME_STEPS * HOP_LENGTH) / SAMPLE_RATE
# Тривалість сегмента в мілісекундах, беремо 95% для запасу
SEGMENT_DURATION_MS = int(SEGMENT_DURATION_SECONDS * 1000 * 0.95)
SEGMENT_OVERLAP_MS = 500  # Перекриття між сегментами в мілісекундах (0.5 секунди)
# Мінімальна тривалість останнього шматка аудіо, щоб його обробляти
MIN_CHUNK_DURATION_MS = int(SEGMENT_DURATION_MS * 0.5)


# --- Функції для роботи з історією аналізів ---
def load_history():
    """Завантажує історію аналізів з JSON-файлу."""
    if not os.path.exists(HISTORY_FILE_PATH):
        return [] # Якщо файлу немає, повертаємо порожній список
    try:
        with open(HISTORY_FILE_PATH, 'r', encoding='utf-8') as f:
            data = json.load(f)
            # Перевірка, чи завантажені дані є списком
            return data if isinstance(data, list) else []
    except json.JSONDecodeError:
        print(f"Попередження: Не вдалося розкодувати JSON з файлу {HISTORY_FILE_PATH}. Повертаю порожню історію.")
        return []
    except Exception as e:
        print(f"Помилка завантаження історії: {e}")
        return []

def save_to_history(record):
    """Зберігає новий запис аналізу в історію (JSON-файл)."""
    history = load_history()
    history.insert(0, record) # Нові записи додаємо на початок
    try:
```

```
        with open(HISTORY_FILE_PATH, 'w', encoding='utf-8') as f:
            json.dump(history, f, ensure_ascii=False, indent=4) # indent для кращої читабельності файлу
    except Exception as e:
        print(f"Помилка збереження історії: {e}")


# --- Визначення класу моделі PyTorch ---
class SpectrogramCNN(nn.Module):
    def __init__(self, num_mels, time_steps, num_classes, dropout_rate=DROPOUT_RATE):
        super(SpectrogramCNN, self).__init__()
        # Перший згортковий блок
        self.conv_block1 = nn.Sequential(
            nn.Conv2d(in_channels=1, out_channels=32, kernel_size=(3, 3), padding='same'),
            nn.ReLU(), nn.BatchNorm2d(32), nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2)))
        # Другий згортковий блок
        self.conv_block2 = nn.Sequential(
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=(3, 3), padding='same'),
            nn.ReLU(), nn.BatchNorm2d(64), nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2)))
        # Третій згортковий блок
        self.conv_block3 = nn.Sequential(
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=(3, 3), padding='same'),
            nn.ReLU(), nn.BatchNorm2d(128), nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2)))
        # Четвертий згортковий блок
        self.conv_block4 = nn.Sequential(
            nn.Conv2d(in_channels=128, out_channels=128, kernel_size=(3, 3), padding='same'),
            nn.ReLU(), nn.BatchNorm2d(128), nn.MaxPool2d(kernel_size=(2, 4))) # Асиметричний пулінг

        # Розрахунок розміру вхідних даних для повнозв'язного шару
        # шляхом пропускання фіктивного тензора через згорткові шари
        with torch.no_grad(): # Градієнти тут не потрібні
            dummy_input = torch.randn(1, 1, num_mels, time_steps)
            dummy_out = self.conv_block1(dummy_input)
            dummy_out = self.conv_block2(dummy_out)
            dummy_out = self.conv_block3(dummy_out)
            dummy_out = self.conv_block4(dummy_out)
            self.flattened_size = dummy_out.view(1, -1).size(1) # Розмір "розгорнутого" тензора

        # Повнозв'язний блок (класифікатор)
        self.fc_block = nn.Sequential(
            nn.Flatten(), # Розгортаємо карти ознак у вектор
            nn.Linear(self.flattened_size, 256), nn.ReLU(),
            nn.Dropout(dropout_rate),
```

```
            nn.Linear(256, num_classes)) # Вихідний шар з кількістю нейронів = кількості класів

    def forward(self, x): # Прямий прохід даних через модель
        x = self.conv_block1(x)
        x = self.conv_block2(x)
        x = self.conv_block3(x)
        x = self.conv_block4(x)
        return self.fc_block(x)


# --- Завантаження навченої моделі та допоміжних файлів ---
# Абсолютний шлях до папки з моделлю (краще використовувати відносні або конфігураційні файли)
MODEL_DIR_API = 'D:/University_2021-2023/8_course/Dyplome/Emotional-detector/backend/backend/model'
MODEL_PATH_API = os.path.join(MODEL_DIR_API, 'model_cnn2d_spectrogram.pth')
ENCODER_PATH_API = os.path.join(MODEL_DIR_API, 'label_encoder_spectrogram.pkl')
SCALER_PATH_API = os.path.join(MODEL_DIR_API, 'spectrogram_scaler.pkl')

try:
    # Завантажуємо кодувальник міток
    with open(ENCODER_PATH_API, 'rb') as f:
        label_encoder = pickle.load(f)
    NUM_CLASSES_API = len(label_encoder.classes_) # Кількість класів емоцій
    print(f"Кодувальник міток завантажено. Класи: {label_encoder.classes_}")

    # Завантажуємо параметри нормалізації (середнє та стандартне відхилення)
    with open(SCALER_PATH_API, 'rb') as f:
        scaler_params = pickle.load(f)
    DATASET_MEAN = scaler_params['mean']
    DATASET_STD = scaler_params['std']
    print(f"Параметри нормалізації завантажено. Середнє: {DATASET_MEAN}, Ст. відхилення: {DATASET_STD}")

    # Ініціалізуємо та завантажуємо ваги моделі
    model_instance = SpectrogramCNN(N_MELS, FIXED_TIME_STEPS, NUM_CLASSES_API, DROPOUT_RATE)
    # map_location='cpu' важливо, якщо модель навчалася на GPU, а використовується на CPU
    model_instance.load_state_dict(torch.load(MODEL_PATH_API, map_location=torch.device('cpu')))
    model_instance.eval() # Переводимо модель у режим оцінки (важливо для Dropout, BatchNorm)
    model = model_instance # Зберігаємо екземпляр моделі для використання
```

```
    print(f"Модель CNN для спектрограм завантажено з {MODEL_PATH_API}")
except FileNotFoundError as e:
    print(f"Помилка: Не знайдено необхідний файл моделі або артефакт: {e}")
    exit() # Завершуємо роботу, якщо не вдалося завантажити модель
except Exception as e:
    print(f"Помилка завантаження моделі або пов'язаних файлів: {e}")
    exit()


# --- Функції для обробки аудіо та генерації спектрограм ---
def audio_to_melspectrogram_api(audio_path, sr=SAMPLE_RATE, n_fft=N_FFT,
hop_length=HOP_LENGTH, n_mels=N_MELS):
    """Перетворює аудіофайл на мел-спектрограму."""
    try:
        y, _ = librosa.load(audio_path, sr=sr) # Завантажуємо аудіо
        # Якщо сигнал дуже тихий, додаємо трохи шуму для стабільності
        if np.abs(y).max() < 1e-5: y += np.random.randn(len(y)) * 1e-6
        # Генеруємо мел-спектрограму
        mel_spec = librosa.feature.melspectrogram(y=y, sr=sr, n_fft=n_fft, hop_length=hop_length,
n_mels=n_mels)
        # Конвертуємо в децибели
        return librosa.power_to_db(mel_spec, ref=np.max)
    except Exception as e:
        print(f"Помилка при генерації мел-спектрограми з {audio_path}: {e}")
        return None

def pad_or_truncate_spectrogram_api(spec, fixed_length=FIXED_TIME_STEPS):
    """Приводить спектрограму до фіксованої довжини (падинг або обрізка)."""
    if spec is None: return None
    current_length = spec.shape[1]
    if current_length < fixed_length:
        # Додаємо нулі справа, якщо спектрограма коротша
        padding_size = fixed_length - current_length
        return np.pad(spec, ((0, 0), (0, padding_size)), mode='constant')
    # Обрізаємо справа, якщо спектрограма довша
    return spec[:, :fixed_length]


# --- Функція для передбачення емоції на одному аудіо сегменті ---
def predict_single_segment_from_audio_obj(audio_segment_obj):
    """Обробляє об'єкт AudioSegment, генерує спектрограму, робить передбачення та
повертає результати."""
    temp_audio_path_segment = None # Ініціалізуємо, щоб уникнути помилки в finally
    try:
```

```
        # Експортуємо сегмент у тимчасовий WAV файл
        with tempfile.NamedTemporaryFile(suffix=".wav", delete=False) as tmp_f:
            audio_segment_obj.export(tmp_f.name, format="wav")
            temp_audio_path_segment = tmp_f.name

        spectrogram_np = audio_to_melspectrogram_api(temp_audio_path_segment)

        if spectrogram_np is None: return None, None, None, None

        processed_spec = pad_or_truncate_spectrogram_api(spectrogram_np)
        if processed_spec is None: return None, None, None, None

        # Нормалізація спектрограми
        normalized_spec = (processed_spec - DATASET_MEAN) / (DATASET_STD + 1e-8) #
Додаємо epsilon для стабільності
        # Перетворення в тензор PyTorch та додавання розмірностей для батчу та каналу
        spec_tensor = torch.tensor(normalized_spec,
dtype=torch.float32).unsqueeze(0).unsqueeze(0)

        # Передбачення моделі
        with torch.no_grad(): # Відключаємо обчислення градієнтів
            prediction_logits = model(spec_tensor)
            probabilities_tensor = F.softmax(prediction_logits, dim=1) # Отримуємо ймовірності
            confidence, predicted_class_index = torch.max(probabilities_tensor, 1) # Найвища
ймовірність та її індекс

        emotion_key = label_encoder.classes_[predicted_class_index.item()] # Ключ емоції
(напр., 'happy')
        # Словник ймовірностей для всіх класів
        probs_dict = {label_encoder.classes_[i]: probabilities_tensor[0][i].item() for i in
range(NUM_CLASSES_API)}

        # Генерація зображення спектрограми (вже нормалізованої) для відповіді API
        plt.figure(figsize=(8, 3)) # Розмір зображення
        librosa.display.specshow(normalized_spec, sr=SAMPLE_RATE,
hop_length=HOP_LENGTH, x_axis='time', y_axis='mel', cmap='viridis')
        plt.colorbar(format='%+2.0f dB') # Колірна шкала
        plt.tight_layout() # Оптимальне розміщення елементів
        img_buffer = io.BytesIO() # Буфер в пам'яті для зображення
        plt.savefig(img_buffer, format='png', bbox_inches='tight', transparent=True) # Зберігаємо
в буфер
        plt.close() # Закриваємо фігуру, щоб не накопичувалися
        img_buffer.seek(0) # Повертаємо курсор на початок буфера
        spectrogram_base64 = base64.b64encode(img_buffer.read()).decode('utf-8') # Кодуємо в
```

```
base64

        return emotion_key, confidence.item(), probs_dict, spectrogram_base64
    except Exception as e:
        print(f"Помилка при передбаченні для сегмента: {e}")
        return None, None, None, None
    finally:
        # Гарантоване видалення тимчасового файлу
        if temp_audio_path_segment and os.path.exists(temp_audio_path_segment):
            os.remove(temp_audio_path_segment)


# --- Створення екземпляру FastAPI ---
app = FastAPI(title="API для Розпізнавання Емоцій (з Сегментацією)")

# Налаштування CORS (Cross-Origin Resource Sharing)
# Дозволяє запити з будь-яких джерел (для розробки, для продакшену краще вказати конкретні)
app.add_middleware(
    CORSMiddleware, allow_origins=["*"], allow_credentials=True,
    allow_methods=["*"], allow_headers=["*"])


# --- Ендпоінт для аналізу аудіо ---
@app.post("/api/audio-analyze", summary="Аналіз емоцій в аудіофайлі (з підтримкою довгих файлів)")
async def analyze_audio_endpoint(file: UploadFile = File(...)):
    temp_audio_path = None # Для гарантованого видалення у випадку помилок на ранніх етапах
    try:
        contents = await file.read() # Читаємо вміст файлу
        audio_io = io.BytesIO(contents) # Створюємо об'єкт BytesIO для pydub

        # Завантажуємо аудіо за допомогою pydub та конвертуємо
        audio_segment_full = AudioSegment.from_file(audio_io)
        audio_segment_full = audio_segment_full.set_channels(1).set_frame_rate(SAMPLE_RATE) # Моно, потрібна частота

        duration_seconds = len(audio_segment_full) / 1000.0 # Тривалість аудіо в секундах
        timestamp_now = datetime.now().isoformat() # Поточний час для запису

        # Початковий вміст відповіді
        response_payload = {
```

```
            "timestamp": timestamp_now,
            "original_filename": file.filename,
            "duration_seconds": round(duration_seconds, 2),
        }

        if duration_seconds <= LONG_AUDIO_THRESHOLD_SECONDS: # Якщо аудіо
КОРОТКЕ
            response_payload["is_long_audio"] = False # Прапорець для фронтенду

            # Обробляємо весь файл як один сегмент
            emotion_key, confidence, probabilities, analyzed_spectrogram_base64 = \
                predict_single_segment_from_audio_obj(audio_segment_full)

            if emotion_key: # Якщо емоцію успішно передбачено
                final_emotion_string = EMOTION_TRANSLATION_MAP.get(emotion_key,
emotion_key)
                translated_probs = {EMOTION_TRANSLATION_MAP.get(k, k): v for k,v in
probabilities.items()}

                # Доповнюємо відповідь
                response_payload.update({
                    "emotion": final_emotion_string,
                    "emotion_key": emotion_key,
                    "confidence": confidence,
                    "probabilities": translated_probs,
                    "analyzed_spectrogram_image":
f"{analyzed_spectrogram_base64}" if analyzed_spectrogram_base64
else None,
                    "example_spectrogram_path":
EXAMPLE_SPECTROGRAM_PATHS.get(emotion_key)
                })
                # Зберігаємо в історію (копію, щоб уникнути випадкових змін)
                history_record_short = response_payload.copy()
                save_to_history(history_record_short)
            else: # Якщо емоцію не вдалося передбачити
                raise HTTPException(status_code=400, detail="Не вдалося передбачити емоцію
для короткого аудіофайлу.")

        else: # Якщо аудіо ДОВГЕ, обробляємо по сегментах
            response_payload["is_long_audio"] = True

            if SEGMENT_DURATION_MS <= 0: # Перевірка на випадок помилки в
конфігурації
                raise HTTPException(status_code=500,
```

```
                    detail="Внутрішня помилка: тривалість сегмента має бути позитивною.")

        # Нарізаємо аудіо на шматки (чанки)
        chunks = make_chunks(audio_segment_full, SEGMENT_DURATION_MS)

        segment_results_list = [] # Список для результатів по кожному сегменту
        all_emotion_keys_detected = [] # Список для визначення переважаючої емоції

        current_time_ms = 0 # Поточний час початку сегмента
        first_segment_spec_b64_for_response = None # Спектрограма першого сегмента для візуалізації

        for i, chunk in enumerate(chunks):
            # Пропускаємо занадто короткі чанки, які можуть утворитися в кінці
            if len(chunk) < MIN_CHUNK_DURATION_MS:
                continue

            # Обробляємо поточний чанк
            emotion_key_chunk, confidence_chunk, probabilities_chunk, spec_b64_chunk = \
                predict_single_segment_from_audio_obj(chunk)

            if i == 0: # Зберігаємо спектрограму першого чанка для відповіді
                first_segment_spec_b64_for_response = spec_b64_chunk

            if emotion_key_chunk: # Якщо емоцію для чанка визначено
                segment_start_sec = current_time_ms / 1000.0
                segment_end_sec = (current_time_ms + len(chunk)) / 1000.0

                # Додаємо результат сегмента до списку
                segment_results_list.append({
                    "start_time_seconds": round(segment_start_sec, 2),
                    "end_time_seconds": round(segment_end_sec, 2),
                    "emotion_key": emotion_key_chunk,
                    "emotion_string": EMOTION_TRANSLATION_MAP.get(emotion_key_chunk, emotion_key_chunk),
                    "confidence": confidence_chunk,
                    "probabilities": {EMOTION_TRANSLATION_MAP.get(k, k): v for k,v in probabilities_chunk.items()}
                })
                all_emotion_keys_detected.append(emotion_key_chunk) # Для визначення переважаючої

            # Оновлюємо час початку наступного сегмента, враховуючи перекриття
```

```
            if SEGMENT_OVERLAP_MS < len(chunk): # Перекриття не може бути більшим за сам чанк
                current_time_ms += (len(chunk) - SEGMENT_OVERLAP_MS)
            else:
                current_time_ms += len(chunk) # Якщо перекриття велике, беремо наступний чанк без пропусків

        # Виводимо для відладки список всіх знайдених ключів емоцій
        print(f"Відладка: Всі ключі емоцій, визначені для довгого аудіо: {all_emotion_keys_detected}")

        if not segment_results_list: # Якщо жоден сегмент не вдалося проаналізувати
            raise HTTPException(status_code=400, detail="Не вдалося проаналізувати жодного сегмента довгого аудіофайлу.")

        # Визначаємо переважаючу емоцію
        predominant_emotion_key = None
        predominant_emotion_string = "Не визначено"
        if all_emotion_keys_detected:
            # Використовуємо Counter для підрахунку частоти кожної емоції
            emotion_counts = Counter(all_emotion_keys_detected)
            if emotion_counts: # Перевірка, чи лічильник не порожній
                predominant_emotion_key = emotion_counts.most_common(1)[0][0]
                predominant_emotion_string = EMOTION_TRANSLATION_MAP.get(predominant_emotion_key, predominant_emotion_key)
            else:
                print("Відладка: Лічильник емоцій порожній, хоча список ключів не порожній.")
        else:
            print("Відладка: Жодної емоції не було визначено в сегментах.")

        # Виводимо для відладки визначену переважаючу емоцію
        print(f"Відладка: Переважаючий ключ емоції: {predominant_emotion_key}")
        print(f"Відладка: Переважаюча емоція (рядок): {predominant_emotion_string}")

        # Доповнюємо відповідь для довгого аудіо
        response_payload.update({
            "segment_analysis": segment_results_list,
            "predominant_emotion": predominant_emotion_string,
            "predominant_emotion_key": predominant_emotion_key,
            "analyzed_spectrogram_image": f"{first_segment_spec_b64_for_response}" if first_segment_spec_b64_for_response else None,
            "example_spectrogram_path":
```

```
EXAMPLE_SPECTROGRAM_PATHS.get(predominant_emotion_key) if
predominant_emotion_key else None
        })

        # Зберігаємо узагальнену інформацію в історію
        history_summary_long = {
            "timestamp": timestamp_now,
            "original_filename": file.filename,
            "duration_seconds": round(duration_seconds, 2),
            "is_long_audio": True,
            "predicted_emotion": predominant_emotion_string,
            "emotion_key": predominant_emotion_key,
            "segment_analysis_summary": f"Проаналізовано {len(segment_results_list)}
сегментів."
        }
        save_to_history(history_summary_long)

    except HTTPException as http_exc: # Перехоплюємо HTTP винятки, щоб вони дійшли до
клієнта
        raise http_exc
    except Exception as e: # Обробка інших непередбачених помилок
        print(f"Критична помилка в ендпоінті /api/audio-analyze: {e}")
        import traceback # Для детального стеку викликів
        traceback.print_exc()
        raise HTTPException(status_code=500, detail=f"Внутрішня помилка сервера при
обробці аудіо: {str(e)}")

    return JSONResponse(content=response_payload) # Повертаємо сформовану відповідь

# --- Ендпоінт для отримання історії аналізів ---
@app.get("/api/history", summary="Отримати історію аналізів")
async def get_analysis_history():
    history = load_history()
    return JSONResponse(content=history)

# --- Базовий ендпоінт для перевірки статусу API ---
@app.get("/", summary="Статус API")
def read_root_cnn():
    return {"message": "API для розпізнавання емоцій (з сегментацією) працює."}

# Для локального запуску: uvicorn main:app --reload --port 8000
```