

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Методи та програмні засоби класифікації музичних творів за жанрами

Виконав: студент II курсу групи 2ПІ- 23м
спеціальності 121 – Інженерія програмного
забезпечення

Каспров Владислав Олександрович

Керівник: к.т.н., доц. каф. ПЗ Ткаченко О.М

Кас « 09 » грудня 2024р.

Опонент: к.т.н., доц. каф. ОТ Тарновський М.Г.

ОТ « 09 » грудня 2024р.

Допущено до захисту

Завідувач кафедри ПЗ

д.т.н., проф. Романюк О. Н.

(прізвище та ініціали)

« 09 » грудня 2024 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., професор Романюк О.Н.

« 18 » вересня 2024 р.

З А В Д А Н Н Я НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Каспрову Владислав Олександровичу

1. Тема роботи – Методи та програмні засоби класифікації музичних творів за жанрами

Керівник роботи: Ткаченко Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 17 » вересня 2024 р. № 310.

2. Строк подання студентом роботи

3 грудня 2024 р.

3. Вихідні дані до роботи: Python 3.10, фреймворки TensorFlow та PyTorch для глибокого навчання, бібліотеки LibROSA та NumPy для обробки аудіоданих, середовище розробки PyCharm, Node.js, React, система контролю версій GitHub, операційна система Windows 10, методи машинного навчання: конволюційні та рекурентні нейронні мережі, механізм уваги (Attention) та адаптивні оптимізатори (Adam)

4. Зміст текстової частини: аналітичний огляд систем та методів класифікації музичних творів за жанровими ознаками; розробка архітектури системи та методів машинного навчання для класифікації творів за жанровими ознаками; програмна реалізація системи та експериментальний аналіз моделі класифікації музичних творів; тестування розробленого програмного продукту; економічна частина

5. Перелік ілюстративного матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ткаченко О.М., к.т.н., доцент кафедри ПЗ	19.09.2024	03.12.2024
5.	Причепя І. В., к.е.н., доцент кафедри ЕПВМ	24.11.2024	30.11.2024

7. Дата видачі завдання 19 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналітичний огляд систем та методів класифікації музичних творів за жанровими ознаками	20.09.2024-30.09.2024	ВЛК
2.	Розробка архітектури системи та методів машинного навчання для класифікації творів за жанровими ознаками	01.10.2024-08.10.2024	ВЛК
3.	Програмна реалізація системи та експериментальний аналіз моделі класифікації музичних творів	09.10.2024-09.11.2024	ВЛК
4.	Тестування розробленого програмного продукту	10.11.2024-24.11.2024	ВЛК
5.	Економічна частина	24.11.2024-30.11.2024	ВЛК

Студент

Керівник магістерської кваліфікаційної


(підпис)

Каспров В.О.
(прізвище та ініціали)
Ткаченко О.М.
(прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Каспров В.О. Методи та програмні засоби класифікації музичних творів за жанрами : магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 154с.

На укр. мові. Бібліогр.: 27 назв; рис.: 18; табл. 20.

У магістерській кваліфікаційній роботі "Методи та програмні засоби класифікації музичних творів за жанрами" розроблено програмне забезпечення, спрямоване на покращення точності та ефективності автоматичної класифікації аудіозаписів музики за жанрами. Дослідження охоплює побудову гібридної моделі глибинного навчання, яка поєднує конволюційні та рекурентні нейронні мережі з механізмом уваги (Attention) для глибокого аналізу спектрально-часових характеристик аудіосигналу. Запропоновано клієнт-серверну архітектуру з вебінтерфейсом, яка забезпечує зручний доступ до функціоналу класифікації та інтеграцію з іншими сервісами через API.

Практична цінність роботи полягає у підвищенні точності та швидкості класифікації музичних творів, що сприяє ефективнішому управлінню великими музичними бібліотеками, покращенню рекомендаційних систем для стрімінгових платформ, а також формуванню персоналізованого користувацького досвіду. Це дає змогу власникам музичних сервісів та контент-агрегаторам оптимізувати пошук та організацію аудіоданих, забезпечуючи більш релевантні рекомендації слухачам.

Ключові слова: класифікація музики, глибинне навчання, конволюційні та рекурентні нейронні мережі, механізм уваги.

ABSTRACT

Kasprov V.O. Methods and Software for Classifying Musical Works by Genre: Master's Thesis in the Specialty 121 – Software Engineering, Educational Program – Software Engineering. Vinnytsia: VNTU, 2024. 154 pages.

In Ukrainian language. Bibliographer: 27 titles; fig.: 18; table. 20.

In the master's qualification work "Methods and Software for Classifying Musical Works by Genre," a software solution has been developed aimed at improving the accuracy and efficiency of automatic classification of music audio recordings by genre. The research encompasses the construction of a hybrid deep learning model that combines convolutional and recurrent neural networks with an attention mechanism to thoroughly analyze the spectral-temporal characteristics of the audio signal. A client-server architecture with a web interface is proposed, ensuring convenient access to classification functionality and integration with other services via an API.

The practical value of the work lies in enhancing the accuracy and speed of classifying musical works, which contributes to more efficient management of large music libraries, improving recommendation systems for streaming platforms, and forming a personalized user experience. This enables music service providers and content aggregators to optimize search and organization of audio data, delivering more relevant recommendations to listeners.

Keywords: music classification, deep learning, convolutional and recurrent neural networks, attention mechanism.

ЗМІСТ

ВСТУП.....	5
1 АНАЛІЗ СИСТЕМ ТА МЕТОДІВ КЛАСИФІКАЦІЇ МУЗИЧНИХ ТВОРІВ ЗА ЖАНРОВИМИ ОЗНАКАМИ	9
1.1 Аналіз предметної області	9
1.2 Дослідження сучасного стану питання	11
1.3 Огляд наявних реалізацій класифікації	13
1.4 Застосування машинного навчання у класифікації.....	19
1.5 Повіряльний аналіз сучасних моделей машинного навчання.....	21
1.6 Аналіз платформ для реалізації класифікаційних систем	23
1.7 Висновки.....	25
2 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ ТВОРІВ ЗА ЖАНРОВИМИ ОЗНАКАМИ.....	27
2.1 Застосування конволюційних нейронних мереж (CNN) для класифікації музики.....	27
2.2 Застосування рекурентних нейронних мереж (RNN) для класифікації музики.....	30
2.3 Застосування механізму уваги (Attention) у класифікації музичних творів.....	31
2.4 Застосування гібридної моделі CNN-RNN-Attention для класифікації музичних творів	34
2.5 Особливості початку гібридної моделі	36
2.6 Розробка архітектури вебсистеми.....	39
2.7 Проектування бази даних користувачів	43
2.8 Удосконалення процедури попередньої обробки аудіосигналу для гібридної моделі.....	45
2.9 Висновки.....	47
3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ТА ЕКСПЕРИМЕНТАЛЬНИЙ АНАЛІЗ МОДЕЛІ КЛАСИФІКАЦІЇ МУЗИЧНИХ ТВОРІВ.....	49
3.1 Вибір мови програмування.....	49
3.2 Вибір середовища програмування	51

3.3 Програмна реалізація конволюційних нейронних мереж (CNN) для класифікації музики.....	53
3.4 Програмна реалізація рекурентних нейронних мереж (RNN) для класифікації музики.....	56
3.5 Програмна реалізація механізму уваги (Attention) у класифікації музичних творів.....	58
3.6 Реалізація клієнт-сервер архітектури	60
3.7 Програмна реалізація удосконалення процедури попередньої обробки аудіосигналу	62
3.8 Програмна реалізація бази даних клієнта	64
3.9 Висновки.....	67
4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ	69
4.1 Аналіз методів тестування.....	69
4.2 Тестування функціональних можливостей програмного продукту	71
4.3 Модульне тестування програмного продукту	74
4.4 Тестування продуктивності та навантаження.....	78
4.5 Тестування клієнтського інтерфейсу.....	80
4.6 Інструкція користувача	84
4.7 Навчання моделі та аналіз результатів	85
4.8 Висновки.....	90
5 ЕКОНОМІЧНА ЧАСТИНА	92
5.1 Проведення комерційного аудиту науково-технічної розробки.....	92
5.2 Розрахунок витрат на здійснення науково-дослідної роботи.....	94
5.2.1 Витрати на оплату праці	95
5.2.2 Відрахування на соціальні витрати.....	98
5.2.3 Сировина та матеріал	99
5.2.4 Розрахунок витрат на комплектуючі	100
5.2.5 Спец устаткування для наукових (експериментальних) робіт	101
5.2.6 Програмне забезпечення для наукових (експериментальних) робіт	102
5.2.7 Амортизація обладнання, програмних засобів та приміщень.....	103
5.2.8 Паливо та енергія для науково-виробничих цілей	105
5.2.9 Службові відрядження	106
5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і	

	4
організації.....	106
5.2.11 Інші витрати	107
5.2.12 Накладні (загальновиробничі) витрати	107
5.3 Оцінювання важливості та наукової значимості науково-дослідної роботи	108
5.4 Висновки.....	114
ВИСНОВКИ	115
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	117
ДОДАТКИ	120
ДОДАТОК А – Технічне завдання.....	121
ДОДАТОК Б – Протокол перевірки МКР на плагіат.....	125
ДОДАТОК В – Лістинг програми.....	126
ДОДАТОК Г – Ілюстративна частина	146

ВСТУП

Обґрунтування вибору теми дослідження

У сучасному цифровому суспільстві музика стала невід'ємною частиною повсякденного життя мільйонів людей. Завдяки розвитку технологій та широкому розповсюдженню інтернету, доступ до музичного контенту значно спростився. Цифрові платформи, стрімінгові сервіси та соціальні мережі щоденно поповнюються тисячами нових музичних творів різноманітних жанрів і стилів. Такий безпрецедентний обсяг інформації створює не лише можливості для слухачів, але й виклики для музичної індустрії щодо ефективної організації, пошуку та рекомендації контенту.

Традиційні методи класифікації музики, засновані на ручному тегуванні та суб'єктивних оцінках експертів, стають неефективними та застарілими в умовах сучасних обсягів даних. Ручне класифікування є трудомістким, затратним за часом і не здатне охопити всю різноманітність нових жанрів та стилів, які постійно з'являються. Крім того, суб'єктивність людського фактору може призводити до неточностей та непослідовностей у класифікації.

У зв'язку з цим виникає нагальна потреба в автоматизованих системах класифікації музичних творів, які здатні швидко та точно визначати жанрову приналежність композицій. Розвиток методів машинного навчання, зокрема глибоких нейронних мереж, відкриває нові перспективи в цьому напрямку. Застосування конволюційних нейронних мереж (CNN) дозволяє ефективно виділяти просторові ознаки з музичного сигналу, такі як тембр та спектральні характеристики. Рекурентні нейронні мережі (RNN) здатні моделювати часові залежності, що є критично важливим для аналізу музичних творів як послідовностей звуків. Додаткове використання механізму уваги (Attention) дозволяє моделі фокусуватися на найбільш релевантних частинах композиції, покращуючи точність та інтерпретованість результатів.

Розробка гібридної моделі, яка поєднує CNN, RNN та механізм уваги, є актуальним та перспективним напрямом досліджень. Така модель здатна

враховувати складні просторово-часові залежності в музичному сигналі, що сприяє підвищенню точності класифікації. Це особливо важливо в контексті сучасної музичної індустрії, де з'являється багато крос-жанрових творів, які важко класифікувати за допомогою традиційних методів.

Крім того, інтеграція розробленої моделі в клієнт-серверну вебсистему з API дозволяє надавати доступ до функціоналу класифікації широкому колу користувачів та розробників. Це відкриває можливості для інтеграції з іншими сервісами, такими як музичні стрімінгові платформи, рекомендаційні системи, додатки для аналізу музики та інші. Такий підхід сприяє підвищенню зручності користування музичними сервісами, покращує якість рекомендацій та задовольняє потреби як кінцевих користувачів, так і бізнесу.

Актуальність теми також обумовлена зростанням конкуренції на ринку музичних сервісів, де швидкість та точність класифікації контенту можуть стати ключовими факторами успіху. Розробка ефективних методів автоматичної класифікації сприяє підвищенню конкурентоспроможності сервісів, дозволяючи їм надавати більш персоналізований та релевантний контент своїм користувачам.

Таким чином, вибір теми дослідження обумовлений як практичними потребами сучасної музичної індустрії, так і науковим інтересом до розвитку методів машинного навчання. Розробка та впровадження ефективної системи автоматичної класифікації музичних творів за жанрами сприятиме вирішенню актуальних проблем організації та доступу до музичного контенту, а також розширить горизонти застосування гібридних нейронних мереж у різних галузях.

Зв'язок роботи з науковими програмами, планами, темами.

Дослідження виконано відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та задачі дослідження. Метою даної магістерської кваліфікаційної роботи є підвищення ймовірності правильних класифікацій музичних творів за рахунок розробки гібридної моделі та вдосконаленню попередньої обробки

аудіосигналу.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- провести аналітичний огляд сучасних методів та систем класифікації музичних творів за жанровими ознаками, визначити їхні переваги та недоліки, а також виявити напрямки для можливого покращення;
- розробити гібридну модель класифікації музичних творів, яка інтегрує різні методи машинного навчання для підвищення точності класифікації;
- вдосконалити метод попередньої обробки аудіосигналу та виділення релевантних ознак;
- розробити клієнт-серверну архітектуру вебсистеми з API, яка забезпечить інтеграцію моделі та надання користувачам доступу до функціоналу автоматичної класифікації музичних творів.
- забезпечити збереження та управління даними за допомогою бази даних, реалізувати логування результатів класифікації та інших важливих подій для подальшого аналізу та вдосконалення системи.

Об'єкт дослідження – процес автоматичної класифікації музичних творів за жанровими ознаками.

Предмет дослідження – методи машинного навчання для класифікації музичних творів, зокрема гібридні моделі, що поєднують конволюційні та рекурентні нейронні мережі з механізмом уваги.

Методи досліджень: методи машинного навчання та глибоких нейронних мереж для розробки класифікаційної моделі, цифрова обробка сигналів для перетворення аудіоданих у спектрограми, методи математичної статистики для оцінювання ефективності розробленої моделі.

Наукова новизна отриманих результатів

1. Запропоновано гібридну модель глибокого навчання для класифікації музичних творів, особливістю якої є поєднання конволюційних та рекурентних нейронних мереж з механізмом уваги для аналізу спектрально-часових ознак аудіосигналу, що дозволило підвищити достовірність та інтерпретованість результатів класифікації.

2. Отримав подальшого розвитку метод попередньої обробки аудіосигналу та виділення релевантних ознак, який відрізняється від існуючих адаптивним вибором спектрально-часових характеристик та фокусуванням уваги на ключових фрагментах сигналу, що дозволило підвищити достовірність класифікації та зменшити час обробки даних.

Практичне значення отриманих результатів.

Розроблено програмний засіб для автоматичної класифікації музичних творів за жанрами, який може бути інтегрований у музичні стрімінгові сервіси, бібліотеки або інші системи управління музичним контентом та підвищує точності класифікації на 3%.

Особистий внесок здобувача.

Всі представлені у магістерській кваліфікаційній роботі наукові результати, отримані автором особисто. Наукові праці опубліковано одноосібно.

Апробація матеріалів магістерської кваліфікаційної роботи.

Результати роботи було представлено на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)»

Публікації.

Результати роботи були опубліковані в матеріалах конференції – «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [1, 2].

1 АНАЛІЗ СИСТЕМ ТА МЕТОДІВ КЛАСИФІКАЦІЇ МУЗИЧНИХ ТВОРІВ ЗА ЖАНРОВИМИ ОЗНАКАМИ

1.1 Аналіз предметної області

Музична індустрія є однією з найдинамічніших і найвпливовіших сфер сучасного мистецтва та культури. Музика виконує важливі функції в житті суспільства, виступаючи засобом комунікації, самовираження та емоційного впливу. Вона відображає культурні, соціальні та історичні особливості різних епох і народів, формуючи унікальну звукову палітру, яка характеризується різноманіттям жанрів і стилів.

Жанрова класифікація музики є фундаментальним аспектом музичної теорії та практики. Жанр визначається як категорія або стиль музичного твору, що відображає сукупність характерних рис, таких як ритм, мелодія, гармонія, тембр, інструментальний склад і форма. Класифікація за жанрами дозволяє структурувати музичний контент, полегшуючи його розуміння, аналіз і сприйняття як музикантами, так і слухачами.

Предметна область автоматичної класифікації музичних творів за жанрами поєднує знання з музикознавства, теорії музики, акустики, інформатики та машинного навчання. Основною метою є розробка систем і алгоритмів, здатних автоматично визначати жанрову приналежність музичного твору на основі його аудіозапису або інших доступних даних.

З музикознавчої точки зору, жанри є історично сформованими категоріями, що відображають еволюцію музичного мистецтва. Кожен жанр має свої особливості, пов'язані з культурним контекстом, естетичними принципами, технікою виконання та композиторськими методами. Наприклад, класична музика характеризується складною гармонією та формою, тоді як рок-музика відзначається енергійним ритмом і використанням електричних інструментів. Розуміння специфіки жанрів важливе для розробки алгоритмів класифікації, оскільки дозволяє визначити ключові ознаки, на які слід звертати увагу при аналізі музичного твору. Це вимагає глибокого вивчення музичної теорії, аналізу

структурних елементів композицій та їхніх інтерпретацій.

З акустичної точки зору [3], музичний сигнал є складним поєднанням звукових хвиль різної частоти та амплітуди. Аналіз акустичних властивостей музичного сигналу дозволяє виявити характерні особливості, притаманні певним жанрам. Це можуть бути спектральні характеристики, темпоральні зміни, тембр інструментів і ритмічні патерни. Акустичний аналіз передбачає використання методів цифрової обробки сигналів для екстракції релевантних ознак з аудіозаписів. Це включає перетворення Фур'є, мел-кепстральні коефіцієнти, хроматичні ознаки та аналіз темпоральних патернів. Ці ознаки слугують вхідними даними для алгоритмів класифікації.

З точки зору інформатики, автоматична класифікація музики є задачею розпізнавання образів або класифікації даних на основі їхніх характеристик. Для її вирішення використовуються алгоритми машинного навчання, які навчаються на підготовлених наборах даних з відомими жанровими мітками. Це можуть бути методи наглядного навчання, де модель вчиться асоціювати певні ознаки з конкретними жанрами. Сучасні підходи використовують глибокі нейронні мережі, здатні автоматично виділяти складні ознаки з даних і виявляти нелінійні залежності. Особливий інтерес викликають конволюційні нейронні мережі для аналізу спектрограм та рекурентні нейронні мережі для моделювання часових залежностей у музичному сигналі.

Психоакустика досліджує сприйняття звуку людиною, включаючи те, як ми інтерпретуємо різні акустичні сигнали як музичні жанри. Сприйняття музики є суб'єктивним процесом, що залежить від культурного контексту, особистого досвіду та емоційного стану слухача. Це ускладнює задачу автоматичної класифікації, оскільки модель повинна враховувати не лише об'єктивні характеристики сигналу, але й аспекти людського сприйняття. Розуміння психоакустичних принципів допомагає розробникам алгоритмів створювати більш ефективні моделі, які краще відображають реальні уподобання та очікування користувачів. Це може включати моделювання сприйняття гучності, висоти тону, тембру та інших параметрів.

Автоматична класифікація музичних творів за жанрами має широке практичне застосування. Вона використовується в музичних стрімінгових сервісах для створення персоналізованих рекомендацій, плейлистів та покращення пошуку. У музичних бібліотеках і архівах вона сприяє організації та каталогізації великих обсягів музичного контенту. У радіомовленні автоматична класифікація дозволяє автоматично підбирати плейлисти відповідно до часу доби, аудиторії або тематичних програм. У музикотерапії вона використовується для підбору музики, що відповідає певним терапевтичним цілям або емоційному стану пацієнта. У маркетингу та рекламі вона допомагає аналізувати тренди та споживчі уподобання, створюючи цільові кампанії.

У контексті стрімкого зростання обсягів музичного контенту та різноманітності жанрів, розробка методів та програмних засобів для класифікації музичних творів за жанрами набуває особливої актуальності.

1.2 Дослідження сучасного стану питання

З розвитком цифрових медіа, стрімінгових сервісів та зростанням обсягу доступного аудіоконтенту потреба в автоматичній класифікації музичних творів стала надзвичайно актуальною. З огляду на значний обсяг музики, який щодня додається до онлайн-бібліотек, стає неможливим виконувати ручну класифікацію всіх музичних композицій. Це спричинило інтерес до автоматизації процесів сортування та тегування музики за жанрами, настроєм, емоційними характеристиками тощо.

Перші спроби автоматичної класифікації музики з'явилися ще на початку 2000-х років, коли з розвитком обчислювальної техніки дослідники почали експериментувати з аналізом акустичних характеристик [4] музичного сигналу. Цей підхід базувався на виділенні ключових ознак музичного твору, таких як спектральні характеристики, темп, ритмічні патерни та тембр. Такі ознаки дозволяли класифікувати композиції за певними критеріями, однак мали значні обмеження. Оскільки музика є складним сигналом з великою кількістю залежностей між різними елементами, простий аналіз основних ознак не завжди

давав задовільні результати.

Окрім проблеми обмеженості класичних методів, варто зазначити, що вручну визначені ознаки не могли охопити всю різноманітність музичних жанрів і стилів, що особливо проявлялося в композиціях, де поєднуються елементи кількох жанрів. Наприклад, в останні десятиліття з'явилася велика кількість композицій, які не можуть бути однозначно віднесені до одного жанру, оскільки вони поєднують елементи електронної музики, джазу, класики або фолку. Це посилює вимоги до класифікаційних систем, які повинні враховувати такі багатожанрові композиції.

Сучасні тенденції в розвитку автоматичної класифікації музики зумовлені потребою в персоналізованих рекомендаціях і можливості швидкого пошуку відповідних композицій для кожного користувача. Музичні стрімінгові сервіси, такі як Spotify, Apple Music, та YouTube Music, активно використовують автоматичну класифікацію для створення персоналізованих плейлистів, рекомендацій, а також для вдосконалення загального користувацького досвіду. Автоматична класифікація дозволяє покращити навігацію та взаємодію з музичною бібліотекою, що є ключовим фактором для комерційного успіху подібних сервісів.

Крім того, технології класифікації музики за жанрами та стилями стали важливим інструментом для музикантів та продюсерів, які прагнуть зрозуміти свою аудиторію і слідкувати за тенденціями ринку. Аналіз жанрової належності композицій дозволяє авторам і продюсерам виявляти стилі, що стають популярними, або, навпаки, знаходяться на межі зникнення. Це допомагає в стратегічному плануванні випуску нових релізів та виявленні потенційно перспективних жанрів, що можуть зацікавити нову аудиторію.

З розвитком сучасних технологій та відкритим доступом до програмних інструментів, таких як Python-бібліотеки для аналізу аудіо, створюються нові можливості для автоматичної класифікації. Дослідники та розробники мають доступ до потужних інструментів, які дозволяють експериментувати з різними підходами до класифікації музики. Наприклад, бібліотеки для роботи з аудіо, такі

як LibROSA, забезпечують прості засоби для попередньої обробки сигналу, виділення ознак і підготовки даних для навчання моделей. Водночас бібліотеки глибокого навчання, такі як TensorFlow та PyTorch, полегшують створення, тренування та тестування складних моделей.

Значну роль у сучасному розвитку автоматичної класифікації відіграє також доступ до великих баз музичних даних, таких як Million Song Dataset. Такі бази надають дослідникам можливість працювати з великими обсягами даних, тестуючи свої моделі в умовах, наближених до реальних. Наявність великих обсягів даних дозволяє не лише підвищити точність класифікації, але й вдосконалити можливості алгоритмів у визначенні рідкісних або складних жанрів.

Сучасні дослідження у галузі автоматичної класифікації музики спрямовані на створення більш точних та ефективних методів, що дозволять зменшити витрати часу і ресурсів на ручне тегування. Крім того, значна увага приділяється питанням адаптації моделей до швидкозмінних тенденцій музичної індустрії, що дозволяє сервісам надавати користувачам релевантні рекомендації та полегшує відкриття нової музики. У перспективі ці технології можуть також знайти застосування в суміжних галузях, таких як музикотерапія, культурні дослідження та освітні програми, що ще раз підкреслює актуальність і важливість розвитку автоматичної класифікації музики.

1.3 Огляд наявних реалізацій класифікації

Автоматична класифікація музики на сьогодні є надзвичайно важливою функцією у сфері цифрової музики. Цей процес дозволяє користувачам завантажувати композиції та отримувати детальний аналіз, що включає не тільки жанрову приналежність, але й оцінку інших характеристик, таких як темп, стиль, ритм, а також емоційна насиченість музики. Проте, навіть найвідоміші платформи, що пропонують функції класифікації музики, часто обмежені певними аспектами, через що неможливо досягти комплексного аналізу. Серед них Spotify, Apple Music, AcousticBrainz і Bandcamp Genre Finder. Далі

розглянемо особливості цих платформ, їхні переваги, недоліки та можливості, щоб зрозуміти, чому необхідне створення покращеної класифікаційної системи, яка б об'єднала найкращі аспекти існуючих рішень.

Spotify [5] (рис 1.1) – це один із найбільш популярних музичних сервісів, який створив унікальну екосистему для користувачів по всьому світу. Запущений у 2008 році, Spotify швидко завоював популярність завдяки доступу до величезної бібліотеки музики та інтуїтивній системі рекомендацій. Основою цієї системи є складні алгоритми машинного навчання, які дозволяють автоматично класифікувати музику на основі жанру, стилю, темпу, енергії та інших характеристик.

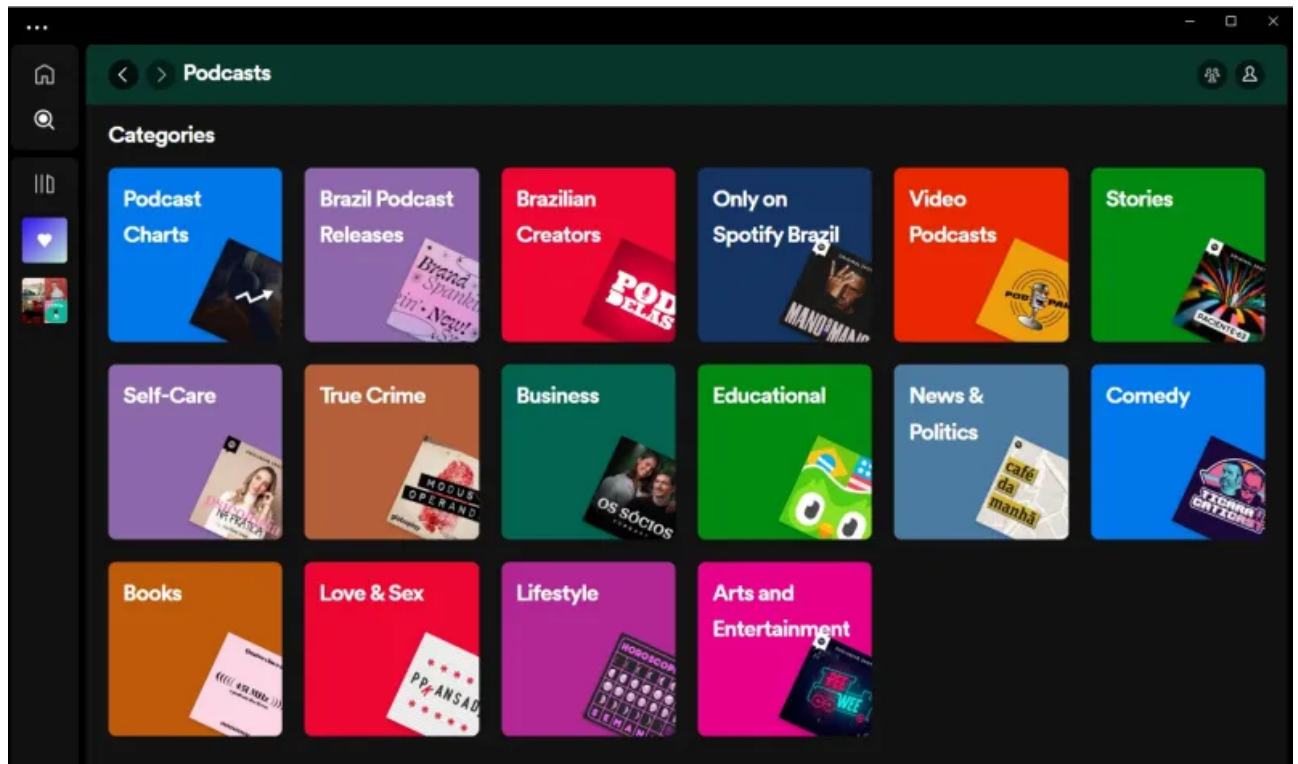


Рисунок 1.1 – Демонстрація роботи Spotify

Алгоритми Spotify аналізують кожну композицію за рядом параметрів, створюючи так звані «аудіо-фінгерпринти» – унікальні профілі треків, що дозволяють зрозуміти загальну структуру та основні особливості композиції. На основі цих даних Spotify пропонує користувачам персоналізовані плейлисти, такі як «Discover Weekly» та «Daily Mix». Ці плейлисти мають значний успіх, адже

дозволяють слухачам відкривати нову музику, не витрачаючи часу на ручний пошук. Проте Spotify орієнтується переважно на комерційну музику і працює з жанровими характеристиками, що є популярними серед масової аудиторії.

Незважаючи на широкі можливості аналізу та доступ до багатого контенту, Spotify має суттєві обмеження, коли мова йде про менш відомі жанри та незалежних виконавців. Алгоритми Spotify налаштовані на роботу з основними музичними стилями та жанрами, а це означає, що аудіофайли, які не вписуються у загальноприйняті категорії, можуть бути класифіковані неточно або взагалі залишитися поза увагою. Це обмежує можливості платформи для користувачів, які шукають специфічні, нетрадиційні або експериментальні жанри музики. Таким чином, Spotify, попри свою популярність і функціональність, не забезпечує повного охоплення потреб слухачів, які цікавляться різноманітною музикою, не лише популярною.

Apple Music [6] (рис. 1.2), запущений у 2015 році, є одним із ключових конкурентів Spotify і займає важливе місце на ринку музичних стрімінгових сервісів. Платформа пропонує унікальний підхід до класифікації музики, комбінуючи автоматизовані алгоритми з кураторськими рекомендаціями. Такий підхід дозволяє Apple Music формувати не лише персоналізовані плейлисти, але й тематичні списки від професійних музичних редакторів, які створюють плейлисти, орієнтовані на певні настрої або події. Крім того, у Apple Music представлені плейлисти, що охоплюють конкретні теми та сезони, що робить платформу привабливою для користувачів, які шукають музику під конкретний настрій.

Однак, як і Spotify, Apple Music зосереджений переважно на комерційній музиці та популярних жанрах. Алгоритми класифікації тут не спрямовані на детальний аналіз аудіо за технічними характеристиками, такими як тембр, енергетика чи динаміка, що було б корисним для користувачів, які хочуть розуміти музику глибше. Apple Music в основному націлений на популярні жанри, через що користувачі, які цікавляться більш експериментальною або незалежною музикою, можуть залишитися незадоволеними. Платформа також

не забезпечує повної автоматизації класифікації, оскільки значну частину контенту курирують редактори, що залишає суб'єктивний слід у класифікації.

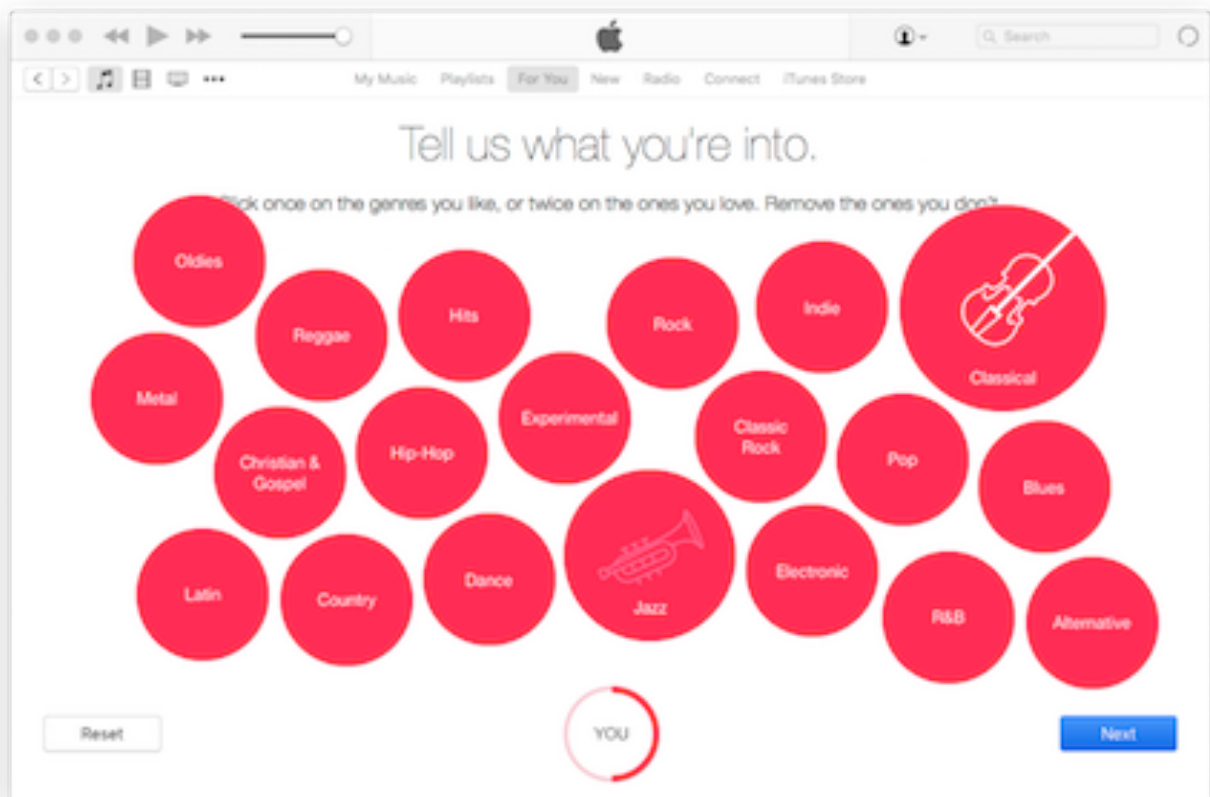


Рисунок 1.2 – Демонстрація роботи Apple Music

AcousticBrainz [7] (рис. 1.3) – це менш відомий, але надзвичайно корисний проєкт із відкритим кодом, який надає детальні метадані про музичні композиції. Створений у співпраці з MusicBrainz, цей проєкт фокусується на детальному технічному аналізі музики, який охоплює такі характеристики, як ритм, темп, динаміка, гармонія, тембр і навіть емоційна насиченість композиції. AcousticBrainz використовує алгоритми машинного навчання для автоматичної класифікації композицій, що робить його надзвичайно цінним для дослідників і розробників, які бажають інтегрувати аудіоаналіз у свої проєкти. Платформа також підтримує API, що дозволяє легко отримувати дані з її бази.

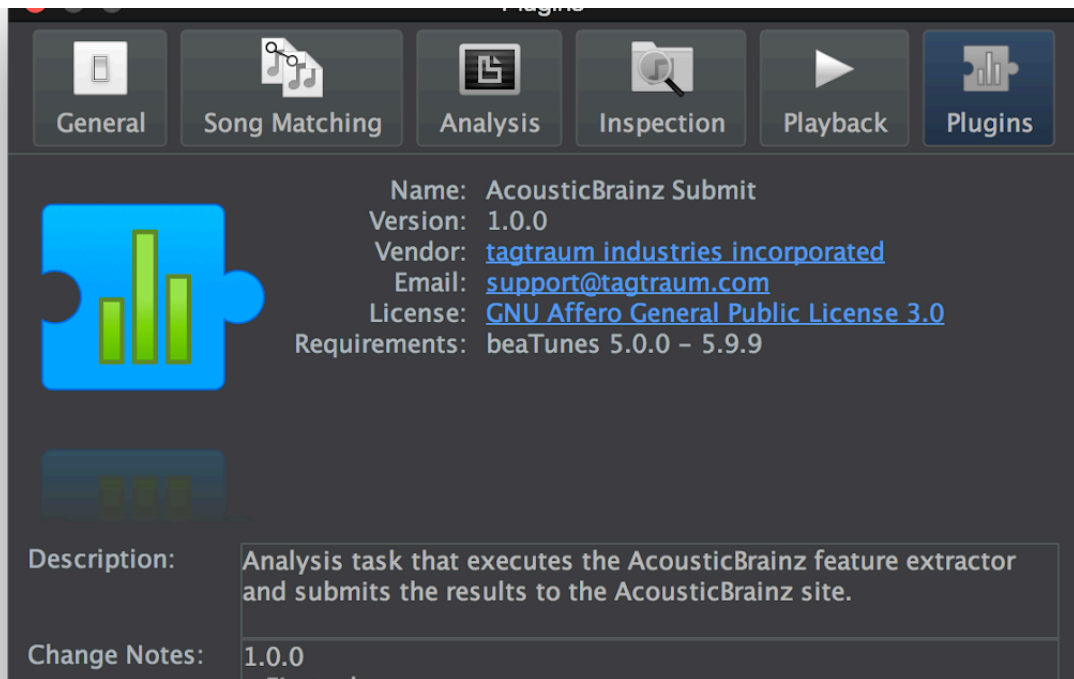


Рисунок 1.3 – Використання плагіну AcousticBrainz

Проте AcousticBrainz не є ідеальним рішенням для кінцевих користувачів. Його інтерфейс і функціональність орієнтовані на технічний аналіз, і платформа здебільшого використовується спеціалістами, які мають досвід роботи з аудіоаналізом. Крім того, AcousticBrainz залежить від даних, наданих спільнотою, що може призводити до неточностей або неповної класифікації, особливо коли йдеться про нові, не дуже популярні жанри. Для звичайного користувача платформа може здатися надто технічною та складною, що робить її менш привабливою для аудиторії, яка не має досвіду роботи з метаданими та аудіоаналізом.

Bandcamp Genre Finder [8] – це інструмент, розроблений для того, щоб допомогти користувачам знаходити музику на основі жанрової класифікації. Bandcamp, будучи платформою для незалежних виконавців, зосереджується на підтримці альтернативної та незалежної музики, що відрізняє його від Spotify та Apple Music. Genre Finder працює на основі метаданих, наданих виконавцями, зокрема тегів, що дає змогу слухачам легко орієнтуватися у великій кількості жанрів. Це важливо для пошуку маловідомих або нішевих стилів, які можуть не входити до основних музичних жанрів.

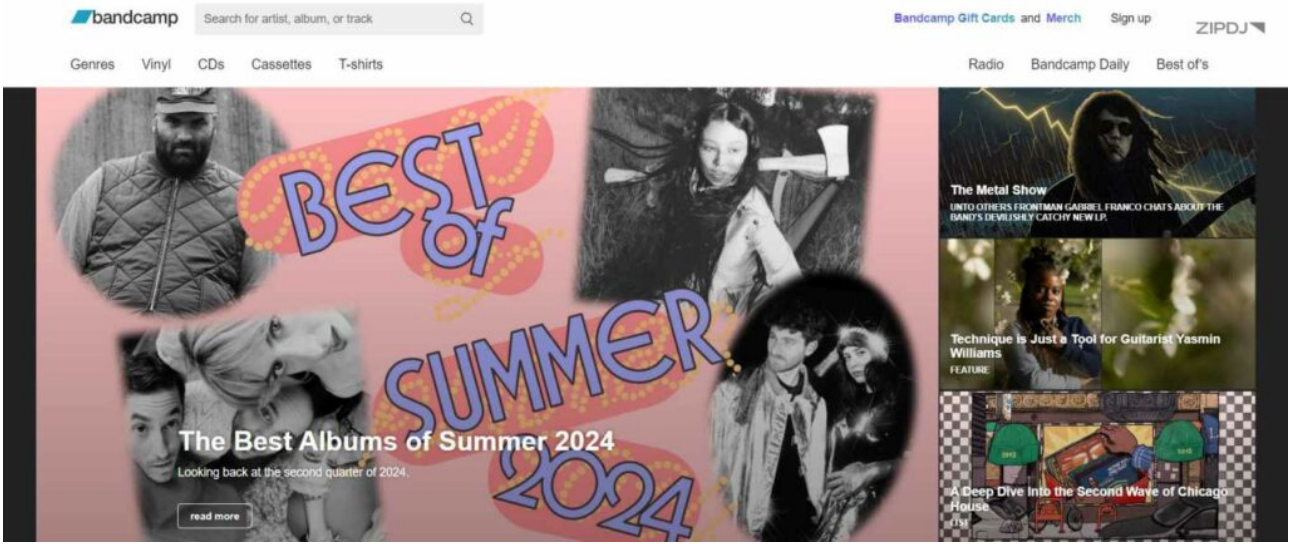


Рисунок 1.4 – Демонстрація роботи Genre Finder

Однак, Genre Finder не виконує автоматичний аудіоаналіз і покладається лише на дані, введені самими виконавцями, що робить процес класифікації суб'єктивним. Часто теги, надані артистами, можуть бути недостатньо детальними або ж узагалі не відповідати реальному жанровому напрямку композиції. Крім того, Genre Finder не надає користувачам доступу до технічних характеристик аудіо, таких як ритм, темп чи тембр, що обмежує можливості для глибшого аналізу. Це обмеження робить інструмент менш універсальним, особливо для тих, хто шукає точного і об'єктивного класифікування.

Таблиця 1.1 – Порівняльний аналіз аналогів

Платформа	Основні функції	Недоліки
Spotify	Персоналізовані рекомендації, аналіз тембру, ритму, енергії	Фокус на комерційну музику, обмежені можливості для класифікації незалежних жанрів
Apple Music	Гібридна система рекомендацій (алгоритми + куратори)	Орієнтація на популярну музику, обмежена класифікація технічних характеристик

Продовження таблиці 1.1

Платформа	Основні функції	Недоліки
AcousticBrainz	Відкритий аудіоаналіз з детальною класифікацією	Відсутність інтерфейсу для кінцевих користувачів, залежність від спільноти, орієнтація на технічний аналіз
Bandcamp Genre Finder	Класифікація за жанрами на основі метаданих	Залежність від введених даних, обмежений аудіоаналіз

Аналіз аналогів (таблиця 1.1) показує, що, незважаючи на інноваційні підходи до класифікації музики, жодна з наявних платформ не задовольняє повністю потреби сучасного користувача. Spotify та Apple Music пропонують інтуїтивні рекомендації та персоналізовані плейлисти, але фокусуються на комерційній музиці, залишаючи без уваги менш популярні жанри. AcousticBrainz пропонує детальний технічний аналіз, проте залишається інструментом для спеціалістів і не має зручного інтерфейсу для звичайних користувачів. Bandcamp Genre Finder орієнтований на підтримку незалежних жанрів, але залежить від суб'єктивних тегів виконавців, що знижує точність класифікації.

Нове програмне рішення могло б поєднати переваги цих платформ, створюючи зручний інтерфейс для широкої аудиторії, але водночас забезпечуючи точний і об'єктивний аналіз за жанрами, стилями та технічними характеристиками.

1.4 Застосування машинного навчання у класифікації

У сучасних умовах автоматична класифікація музики вимагає ефективних інструментів, здатних обробляти значні обсяги даних та враховувати різноманітні характеристики музичних творів. Застосування методів машинного

навчання в цій сфері стало одним з найбільш перспективних підходів, що дозволяє підвищити точність класифікації, адаптуватися до змін у жанрах та стилях музики, а також автоматизувати процес маркування та тегування.

Основна концепція [9] використання машинного навчання для класифікації музичних творів базується на можливості алгоритмів самостійно виявляти структури в даних, що відображають особливості жанру чи стилю. Це досягається за рахунок навчання моделей на великих обсягах даних, що містять різні музичні жанри. При цьому моделі можуть враховувати не лише базові ознаки музичного сигналу, такі як ритм або темп, але й більш комплексні особливості, які відображають гармонічні та мелодійні структури композицій.

На ранніх етапах розвитку автоматичної класифікації дослідники використовували класичні алгоритми машинного навчання, зокрема метод опорних векторів, метод найближчих сусідів, випадкові ліси та наївні баєсові класифікатори. Ці алгоритми дозволяли аналізувати ознаки музичного сигналу, проте вимагали вручну визначених характеристик для навчання моделі. Таким чином, до кожної музичної композиції додавалися ознаки, такі як спектральна щільність, темпоральні зміни або ритмічні структури, які використовувалися як основа для прийняття рішення щодо жанрової належності твору. Проте, ручний вибір характеристик мав низку обмежень, оскільки не дозволяв враховувати складні взаємозв'язки між елементами музики, що часто зустрічаються у багатожанрових композиціях.

Пізніше, із розвитком алгоритмів глибокого навчання, можливості автоматичної класифікації суттєво розширилися. Завдяки появі нейронних мереж з великою кількістю шарів, таких як глибокі нейронні мережі, стало можливим відмовитися від необхідності ручного визначення ознак. Натомість такі моделі здатні самостійно виділяти та аналізувати важливі характеристики музичного сигналу, що робить їх більш точними та універсальними для роботи з широким спектром музичних жанрів.

Одним з ключових методів для обробки музичних даних у класифікації є спектрограми, які представляють частотний спектр аудіосигналу в часі.

Спектрограма перетворює звук в зображення, яке може бути використане як вхідні дані для моделі машинного навчання. Це дозволяє ефективніше аналізувати музичний сигнал, оскільки спектральні ознаки здатні відобразити тембр, ритм та інші особливості композиції. Застосування спектрограм є основою для подальшого використання складних моделей, таких як конволюційні нейронні мережі, які здатні виявляти закономірності у двовимірних представленнях даних.

Окрім спектрограм, у класифікації музичних творів часто використовуються інші методи обробки аудіо, такі як аналіз мел-кепстральних коефіцієнтів (MFCC) та гармонічно-перкусивна декомпозиція (HPSS). MFCC дозволяють виділити основні ознаки звуку, що відповідають сприйняттю людського вуха, і таким чином відображають ключові характеристики тембру та ритму. Гармонічно-перкусивна декомпозиція, своєю чергою, розділяє аудіосигнал на дві складові: гармонічну, що відповідає за мелодію, і перкусивну, яка відображає ритмічні особливості..

Таким чином, застосування машинного навчання у класифікації музики є важливим і перспективним напрямом, який значно розширив можливості аналізу музичних даних та дозволив підвищити точність автоматичного визначення жанрів. Це сприяє розвитку музичних платформ і відповідає потребам сучасного суспільства, де великий обсяг даних вимагає ефективних та автоматизованих рішень для їхньої обробки.

1.5 Повітряльний аналіз сучасних моделей машинного навчання

Для автоматичної класифікації музики сьогодні існує чимало підходів, кожен з яких має свої переваги та обмеження. Сучасні моделі машинного навчання розрізняються за своїми характеристиками та тим, як саме вони аналізують музичний сигнал, що дозволяє вибирати оптимальні варіанти залежно від конкретної задачі. У таблиці 1.2 подано короткий огляд основних моделей, які активно застосовуються для класифікації музичних творів.

Таблиця 1.2 – Огляд основних машинних моделей для класифікації

Модель	Основні характеристики	Переваги	Обмеження
Конволюційні нейронні мережі (CNN)	Використовуються для аналізу спектрограм як двовимірних зображень, дозволяють виявляти локальні патерни в музичних даних	Висока точність у виділенні тембральних і ритмічних характеристик	Не здатні повністю враховувати довготривалі часові залежності
Рекурентні нейронні мережі (RNN)	Використовуються для обробки послідовностей даних; модифікації LSTM і GRU можуть зберігати інформацію на тривалих інтервалах часу	Ефективні для аналізу послідовностей, дозволяють моделювати часову структуру музичних творів	Проблеми з продуктивністю на довгих послідовностях, обмежена ефективність для складних багатожанрових композицій
Механізм уваги (Attention)	Використовує увагу для виділення ключових частин послідовності, фокусуючись на найважливіших елементах композиції	Ефективно працює з довгими послідовностями, враховує глобальний контекст	Часто потребує комбінування з іншими методами для досягнення максимальної ефективності
Варіаційні автокодувальники (VAE)	Зменшують розмірність даних та виділяють приховані ознаки, можуть створювати стислу репрезентацію музичних творів	Можливість генерувати нові зразки для тренування, підвищення точності в класифікації рідкісних жанрів	Переважно використовуються як додатковий інструмент, не завжди є достатньо ефективними як самостійна модель

Поєднання CNN, RNN та механізму уваги дозволить створити більш гнучку і точну систему для класифікації музики. Гібридна модель здатна адаптуватися до різноманітних музичних стилів і жанрів, обробляти багатопланові композиції, що містять різні елементи жанрів, і забезпечувати високу продуктивність завдяки інноваційній архітектурі. Така система дозволить досягти значно вищої точності в класифікації, оскільки вона об'єднує сильні сторони кожного методу, мінімізуючи їхні обмеження.

Таким чином, створення гібридної моделі CNN-RNN-Attention є обґрунтованим вибором для задачі автоматичної класифікації музичних творів, адже вона здатна забезпечити надійну й ефективну обробку складних музичних даних і відповідає вимогам сучасної музичної індустрії.

1.6 Аналіз платформ для реалізації класифікаційних систем

При розробці системи для автоматичної класифікації музики важливим є вибір платформи, яка не лише забезпечить надійний доступ до моделі, але й дозволить легко інтегрувати її у зовнішні застосунки та надати користувачам зручний інтерфейс. На ринку існують різні підходи до реалізації подібних систем: це можуть бути мобільні додатки, десктопні програми, або ж вебсайти, доступні через API. У кожного з них є свої переваги та обмеження, детальніше це зображено у таблиці 1.3.

Таблиця 1.3 – Огляд платформ для реалізації

Платформа	Переваги	Обмеження
Мобільний додаток	Постійний доступ для користувача, зручний інтерфейс	Обмеження потужності мобільних пристроїв, складність адаптації для різних операційних систем

Продовження таблиці 1.3

Десктопний застосунок	Можливість обробки великих обсягів даних на комп'ютері користувача	Прив'язка до конкретного пристрою, складність забезпечення доступності з різних пристроїв
Вебсайт з API	Доступ з будь-якого пристрою з інтернетом, легка інтеграція з іншими додатками, централізоване оновлення	Потреба в інтернет-з'єднанні, вимога забезпечення надійної серверної інфраструктури

Мобільні додатки можуть забезпечити зручний інтерфейс для користувачів, а також мають перевагу в тому, що вони постійно доступні в кишені користувача. Однак розробка мобільних застосунків вимагає адаптації до різних операційних систем (Android, iOS), що значно збільшує складність проекту. Крім того, мобільні додатки обмежені потужністю обчислень на пристроях, що ускладнює роботу з нейронними мережами, які потребують значних ресурсів для класифікації. Тому мобільні додатки менш підходять для глибоких класифікаційних моделей, особливо якщо система обробляє великі обсяги аудіоданих.

Десктопні застосунки також можуть запропонувати користувачам зручний інтерфейс і мають більші можливості обробки даних порівняно з мобільними. Вони дозволяють запускати більш складні алгоритми і здійснювати аналіз безпосередньо на комп'ютері користувача. Однак, десктопні програми мають обмеження в тому, що вони прив'язані до конкретного пристрою, і доступ до них можливий лише з комп'ютера, де встановлено програму. Це знижує гнучкість і доступність системи для широкого кола користувачів, оскільки десктопний застосунок не завжди зручний і може мати обмеження для віддаленого доступу.

Вибір вебсайту з доступом до класифікаційної системи через API є

оптимальним варіантом. Вебсайт забезпечує доступ до системи з будь-якого пристрою, що має інтернет-з'єднання, незалежно від операційної системи чи типу пристрою. Крім того, API створює інтерфейс, через який інші додатки можуть інтегрувати класифікаційну модель у свої сервіси. Така архітектура дозволяє використовувати потужні серверні обчислення, забезпечуючи високу продуктивність і стабільність навіть при великих обсягах даних. API також спрощує процес оновлення моделі та її вдосконалення без потреби в переналаштуванні або перевстановленні застосунку користувачами.

1.7 Висновки

Проведений аналіз предметної області показав, що музична індустрія є складною та багатогранною сферою, де жанрова класифікація музики відіграє ключову роль у структуризації та сприйнятті музичного контенту. Автоматична класифікація музичних творів поєднує знання з музикознавства, акустики, інформатики та машинного навчання, що дозволяє створювати системи, здатні аналізувати музичний сигнал і визначати його жанрову приналежність.

Сучасний стан питання свідчить про зростаючу потребу в автоматизації процесів класифікації музики через значний обсяг доступного аудіоконтенту та різноманітність жанрів. Традиційні методи класифікації виявилися недостатньо ефективними для роботи з багатожанровими композиціями та швидкозмінними тенденціями музичної індустрії.

Аналіз існуючих реалізацій класифікації музики, таких як Spotify, Apple Music, AcousticBrainz та Bandcamp Genre Finder, виявив, що жодна з цих платформ не задовольняє повністю потреби сучасного користувача. Популярні стрімінгові сервіси фокусуються на комерційній музиці та основних жанрах, залишаючи поза увагою незалежні та менш популярні стилі. Інструменти для технічного аналізу, як-от AcousticBrainz, орієнтовані на спеціалістів і не мають зручного інтерфейсу для широкої аудиторії.

Застосування машинного навчання, особливо методів глибокого навчання, відкриває нові можливості для автоматичної класифікації музики. Використання

конволюційних нейронних мереж (CNN) дозволяє ефективно виділяти тембральні та ритмічні характеристики з музичного сигналу. Рекурентні нейронні мережі (RNN) здатні моделювати часові залежності та аналізувати послідовності даних, що важливо для розуміння структури композиції. Механізм уваги (Attention) додає можливість фокусуватися на ключових елементах музичного твору, що покращує точність класифікації складних багатожанрових композицій.

Порівняльний аналіз сучасних моделей машинного навчання показав, що поєднання CNN, RNN та механізму уваги є перспективним підходом для створення гібридної моделі класифікації. Така модель здатна враховувати як просторові, так і часові характеристики музики, забезпечуючи високу точність та гнучкість у роботі з різними жанрами та стилями.

Аналіз платформ для реалізації класифікаційних систем продемонстрував, що вибір вебсайту з доступом до класифікаційної моделі через API є оптимальним рішенням. Це забезпечує доступність сервісу з будь-якого пристрою з інтернет-з'єднанням, легку інтеграцію з іншими додатками та можливість централізованого оновлення моделі без необхідності вносити зміни на стороні користувача.

На основі проведеного дослідження можна зробити висновок, що розробка гібридної моделі класифікації музики, яка поєднує CNN, RNN та механізм уваги, з реалізацією у вигляді вебсервісу з API, є актуальним та обґрунтованим напрямом. Такий підхід дозволить подолати обмеження існуючих рішень, забезпечить точну та об'єктивну класифікацію музичних творів, а також надасть зручний інтерфейс для широкого кола користувачів. Це відповідає сучасним потребам музичної індустрії та сприятиме подальшому розвитку автоматизації в сфері музичного аналізу.

2 РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ ТА МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ ТВОРІВ ЗА ЖАНРОВИМИ ОЗНАКАМИ

2.1 Застосування конволюційних нейронних мереж (CNN) для класифікації музики

Конволюційні нейронні мережі (CNN) [11] стали справжнім проривом у світі машинного навчання. Хоча вони спочатку були розроблені для задач комп'ютерного зору, їхня ефективність не обмежується лише обробкою зображень. Сьогодні CNN успішно застосовуються для аналізу аудіосигналів, зокрема в класифікації музичних творів. Їхня здатність автоматично виділяти релевантні ознаки з вхідних даних робить їх незамінними при роботі зі складними та високорозмірними даними, такими як музичні сигнали.

Але як же застосувати CNN до музики, яка за своєю природою є одновимірним аудіосигналом? Виявляється, цей сигнал можна перетворити у двовимірне представлення, інтерпретуючи його як зображення. Це досягається шляхом обчислення спектрограми або мел-спектрограми, які відображають залежність амплітуди або потужності сигналу від часу та частоти.

Ключовим інструментом для цього є короткочасне перетворення Фур'є (STFT), яке дозволяє розкласти сигнал на його частотні компоненти в різні моменти часу. Формула STFT виглядає наступним чином:

$$X(t, f) = \sum_{n=0}^{N-1} x[n]w[n - t]e^{-j2\pi fn/N} \quad (2.1)$$

де:

$x[n]$ — дискретний аудіосигнал;

$w[n]$ — віконна функція (наприклад, вікно Гаусса або Хеннінга);

N — довжина вікна;

t — час;

f — частота.

Таким чином, ми отримуємо двовимірне зображення — спектрограму, де по одній осі відкладається час, а по іншій — частота, а значення пікселів відповідають амплітудам сигналу. Це дозволяє використовувати CNN для аналізу музики так само, як вони аналізують зображення.

Конволюційна нейронна мережа складається з послідовності шарів, кожен з яких виконує певну роль. Перш за все, це конволюційні шари, які виконують операцію згортки між вхідними даними та наборами фільтрів, виділяючи локальні ознаки. Операція згортки математично описується формулою:

$$(X * W)(i, j) = \sum_m \sum_n X(i - m, j - n) * W(m, n) \quad (2.2)$$

де:

XX — вхідне зображення (спектрограма);

WW — ядро згортки (фільтр);

i, j, i, j — координати пікселя в результаті згортки.

Уявімо, що фільтр розміром 3×3 "проводить" по спектрограмі та виявляє певні патерни, наприклад, характерні для певних частотних компонентів музичного жанру.

Далі йдуть функції активації, які додають нелінійність до моделі. Найпоширенішою є функція ReLU (Rectified Linear Unit), яка визначається як:

$$ReLU(x) = \max(0, x) \quad (2.3)$$

Ця проста, але ефективна функція "відсікає" негативні значення, залишаючи лише позитивні, що допомагає моделі навчатися більш складним залежностям.

Після конволюційних шарів часто застосовуються пулінгові шари, які зменшують розмірність даних, зберігаючи найважливіші ознаки. Наприклад, Max Pooling вибирає максимальне значення в певній області, що дозволяє

зменшити розмір зображення, зберігаючи найбільш виразні ознаки.

Нарешті, повнозв'язані шари виконують кінцеву класифікацію на основі виділених ознак. Вони об'єднують інформацію з попередніх шарів і прогнозують, до якого класу (жанру) належить музичний твір.

Щоб краще зрозуміти, як це працює, розглянемо практичний приклад. Припустимо, ми маємо аудіозапис тривалістю 30 секунд. Використовуючи STFT, ми перетворюємо його в спектрограму розміром 300x300 пікселів. Ця спектрограма стає вхідними даними для нашої CNN.

Перший конволюційний шар може мати, скажімо, 32 фільтри розміром 3x3, які виявляють прості ознаки, такі як горизонтальні або вертикальні лінії, що відповідають стабільним частотам або швидким змінам. Наступні шари можуть виділяти більш складні патерни, комбінуючи ознаки з попередніх шарів.

В результаті, модель навчається розпізнавати характерні спектральні особливості, притаманні певним музичним жанрам. Наприклад, у рок-музиці можуть домінувати гітари з характерним спектром, тоді як у електронній музиці – синтезовані звуки з іншими частотними компонентами.

Використання CNN дозволяє досягти високої точності класифікації. Однак, варто пам'ятати про певні обмеження. Музичні сигнали можуть бути доволі довгими, що призводить до великих розмірів спектрограм і, відповідно, потребує значних обчислювальних ресурсів. Крім того, CNN не враховують часові залежності в повній мірі, оскільки фокусуються на локальних ознаках.

Тому для покращення результатів часто поєднують CNN з іншими методами, такими як рекурентні нейронні мережі (RNN), які добре працюють з послідовними даними, або механізм уваги (Attention), який дозволяє моделі фокусуватися на найбільш важливих частинах сигналу.

У підсумку, конволюційні нейронні мережі відкривають широкі можливості для автоматичної класифікації музики. Їхня здатність виділяти складні ознаки з аудіосигналу робить їх потужним інструментом у цій галузі. системи, що відповідають вимогам сучасної музичної індустрії.

2.2 Застосування рекурентних нейронних мереж (RNN) для класифікації музики

Рекурентні нейронні мережі [12] (RNN) є потужним інструментом для обробки послідовних даних, що робить їх особливо корисними для задач класифікації музичних творів. Музика за своєю природою є послідовністю звуків, які розвиваються в часі, тому моделі, здатні враховувати часові залежності, мають перевагу в аналізі таких даних.

На відміну від звичайних нейронних мереж, RNN мають зворотні зв'язки, що дозволяє їм "пам'ятати" інформацію з попередніх кроків. Це досягається шляхом введення прихованого стану h_t , який залежить від вхідних даних x_t та попереднього прихованого стану h_{t-1} :

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.4)$$

де:

ϕ – нелінійна функція активації (наприклад, $\tanh$);

W_{xh} та W_{hh} – матриці ваг;

b_h – вектор зміщень.

Вихід мережі y_t визначається як:

$$y_t = \sigma(W_{hy}h_t + b_y) \quad (2.5)$$

де σ — функція активації (наприклад, softmax для класифікації).

Для того щоб навчити RNN ефективно розпізнавати музичні жанри, використовується метод зворотного поширення помилки в часі (Backpropagation Through Time, BPTT). Цей метод дозволяє обчислити похибку на кожному часовому кроці та оновити ваги мережі з урахуванням всіх попередніх станів. Функція втрат у цьому випадку зазвичай обирається як крос-ентропія:

$$L = - \sum_{t=1}^T \sum_{k=1}^K y_{tk} \log(\hat{y}_{tk}) \quad (2.6)$$

де:

T – кількість часових кроків;

K – кількість класів;

y_{tk} – істинна мітка класу k у момент часу t ;

$\hat{y}_{tk}$ – прогнозована ймовірність приналежності до класу k у момент часу t .

Рекурентні нейронні мережі є надзвичайно потужним інструментом для класифікації музики завдяки їхній здатності враховувати часові залежності в аудіосигналі. Їхнє поєднання з іншими сучасними методами машинного навчання, такими як CNN та механізм уваги, дозволяє створювати ефективні та гнучкі системи. Ці гібридні моделі підвищують точність класифікації та відкривають нові можливості в аналізі музичних творів, що є надзвичайно актуальним у сучасному світі цифрової музики.

2.3 Застосування механізму уваги (Attention) у класифікації музичних творів

Механізм уваги [13] (Attention) став справжнім проривом у сфері машинного навчання, відкриваючи нові горизонти в обробці послідовних даних. У контексті класифікації музичних творів цей підхід дозволяє моделям зосереджуватися на найбільш важливих частинах аудіосигналу, що мають вирішальне значення для правильного визначення жанру або стилю композиції.

Уявімо, що ми працюємо з послідовністю векторів ознак $X = [x_1, x_2, \dots, x_T]$, де T – довжина аудіозапису. Кожен вектор x_t представляє набір характеристик музичного сигналу в момент часу t . Завдання моделі полягає в тому, щоб визначити, які з цих моментів є найбільш релевантними для класифікації.

Для цього вводиться функція оцінки важливості, яка для кожного елемента x_t обчислює скалярне значення e_t :

$$e_t = \text{score}(x_t, s) \quad (2.7)$$

де:

s — контекстний вектор або прихований стан моделі;

score — функція, яка може бути реалізована різними способами, наприклад, як скалярний добуток або невелика нейронна мережа.

Ці оцінки відображають, наскільки важливим є кожен момент часу для поточного завдання.

Потім ці оцінки нормалізуються за допомогою softmax-функції, щоб отримати ваги уваги α_t :

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^T \exp(e_k)} \quad (2.8)$$

Ці ваги мають властивість сумуватися до 1, що інтерпретується як ймовірність або ступінь уваги, яку модель приділяє кожному елементу послідовності. Після цього обчислюється контекстний вектор c , який є зваженою сумою вхідних ознак:

$$c = \sum_{t=1}^T \alpha_t x_t \quad (2.9)$$

Цей контекстний вектор містить агреговану інформацію про найбільш значущі частини аудіосигналу і використовується для подальшої класифікації. Наприклад, він може бути поданий на вхід повнозв'язаного шару з функцією softmax для передбачення ймовірностей належності до різних жанрів.

Функція втрат для навчання моделі зазвичай вибирається як крос-ентропія:

$$L = - \sum_{t=1}^T \sum_{k=1}^K y_{tk} \log(\hat{y}_{tk}) \quad (2.10)$$

де:

N — кількість зразків у навчальному наборі;

K — кількість класів (жанрів);
 y_{ik} — істинна мітка класу;
 $\hat{y}_{tk}$ — прогнозована ймовірність.

Використання механізму уваги має кілька важливих переваг. По-перше, модель стає здатною адаптивно визначати, які частини аудіосигналу є найбільш релевантними для класифікації. Це особливо цінно для музичних творів зі складною структурою, де важливі елементи можуть бути розподілені нерівномірно по всій тривалості треку. По-друге, механізм уваги сприяє поліпшенню точності моделі, оскільки зменшується вплив нерелевантної або зашумленої інформації. По-третє, ваги уваги a_t надають можливість інтерпретувати роботу моделі, оскільки вони вказують на те, які саме фрагменти аудіо найбільше вплинули на кінцеве рішення.

Розглянемо приклад використання механізму уваги в моделі для класифікації музики. Нехай ми маємо модель на основі рекурентної нейронної мережі (RNN), яка обробляє послідовність ознак x_t і генерує приховані стани h_t :

$$h_t = \text{RNNCell}(x_t, h_{t-1}) \quad (2.11)$$

Функція оцінки важливості може бути реалізована як адитивна увага:

$$e_t = v \tanh(Wh_t + b) \quad (2.12)$$

де W, v, b — параметри, що навчаються. Після обчислення ваг уваги a_i і контекстного вектора c , модель робить передбачення:

$$\hat{y} = \text{softmax}(W_c c + b_c) \quad (2.13)$$

де W_c, b_c — параметри вихідного шару.

Одним із викликів при використанні механізму уваги є підвищена обчислювальна складність, особливо для довгих послідовностей. Це може вимагати оптимізації коду та використання потужних апаратних ресурсів. Однак, завдяки сучасним технологіям та графічним процесорам (GPU), ці проблеми можна ефективно вирішувати.

2.4 Застосування гібридної моделі CNN-RNN-Attention для класифікації музичних творів

Поєднання конволюційних нейронних мереж (CNN), рекурентних нейронних мереж (RNN) та механізму уваги (Attention) у гібридну модель є ключовим етапом у створенні високоефективних систем класифікації музичних творів. Цей інтегрований підхід дозволяє максимально використати сильні сторони кожної окремої технології, забезпечуючи глибокий та всебічний аналіз аудіосигналів. У цьому розділі розглянемо, як саме було здійснено інтеграцію цих компонентів, а також наведемо приклади схематичних зображень, які ілюструють архітектуру моделі.

Першим кроком у створенні гібридної моделі було перетворення сирого аудіосигналу у формат, придатний для аналізу конволюційними нейронними мережами. Використовуючи короткочасне перетворення Фур'є (STFT) (формула 2.1), аудіосигнал було перетворено на спектрограму, яка відображає залежність амплітуди сигналу від часу та частоти. Отримана спектрограма служить вхідними даними для CNN, яка виконує операції згортки для виділення локальних ознак, таких як частотні патерни та тембральні характеристики, характерні для різних музичних жанрів.

Конволюційна нейронна мережа складається з кількох послідовних шарів, кожен з яких виконує певну функцію. Початкові конволюційні шари використовують фільтри для виявлення просторових патернів у спектрограмі. Операція згортки математично описується формулою 2.2.

Після конволюційних шарів застосовуються функції активації, такі як ReLU (Rectified Linear Unit), яка визначається за формулою 2.3.

Ця функція додає нелінійність до моделі, що дозволяє їй навчатися більш складним залежностям. Після декількох шарів згортки та пулінгу, які зменшують розмірність даних, отримані ознаки розгортаються у послідовність для подальшої обробки рекурентною нейронною мережею.

Рекурентні нейронні мережі використовуються для моделювання часових залежностей між послідовними фреймами спектрограми. Прихований стан RNN та вихід мережі обчислюється за формулами 2.4 та 2.5.

Наступним етапом інтеграції було додавання механізму уваги (Attention), який дозволяє моделі фокусуватися на найбільш релевантних частинах послідовності.

Для кращого розуміння інтеграції цих компонентів, можна розглянути рисунок 2.1.

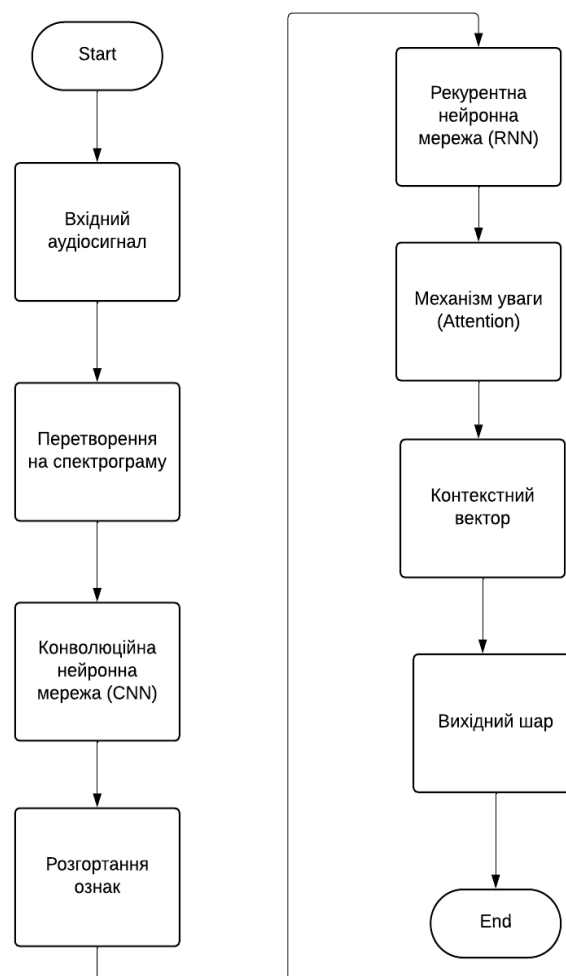


Рисунок 2.1 – Архітектура гібридної моделі

Для наочності можна розглянути конкретний приклад роботи гібридної моделі. Припустимо, ми маємо аудіозапис тривалістю 30 секунд, який ми хочемо класифікувати як рок, джаз або класичну музику. Спочатку аудіозапис перетворюється на спектрограму за допомогою STFT. Потім ця спектрограма проходить через кілька шарів CNN, які виділяють важливі частотні компоненти та тембральні особливості, характерні для різних жанрів. Вихід CNN розгортається у послідовність ознак, яка подається на вхід RNN. RNN моделює часові залежності між цими ознаками, визначаючи, як вони розвиваються протягом тривалості композиції.

Механізм уваги застосовується до прихованих станів RNN, визначаючи, які частини послідовності є найбільш важливими для класифікації. Наприклад, у рок-музиці важливими можуть бути частини з активними гітарними рифами, тоді як у джазі — імпровізаційні соло. Ваги уваги визначають, наскільки важливим є кожен момент часу для остаточного рішення моделі. На основі цих ваг формується контекстний вектор, який подається на вихідний шар для передбачення жанру композиції.

Загалом, інтеграція CNN, RNN та механізму уваги у гібридну модель створює потужний інструмент для класифікації музики, який поєднує у собі найкращі властивості кожної з окремих технологій. Це дозволяє створювати моделі, які є не лише точними, але й гнучкими, здатними адаптуватися до різних музичних стилів та жанрів, що робить їх надзвичайно корисними у сучасній музичній індустрії.

2.5 Особливості навчання гібридної моделі

Навчання гібридної моделі [14], яка поєднує конволюційні нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та механізм уваги, є складним процесом, що вимагає ретельного підходу та врахування специфічних особливостей. Ця складність пов'язана з необхідністю оптимізації великої кількості параметрів та забезпечення узгодженої роботи різних компонентів моделі. При навчанні такої моделі важливо не лише правильно налаштувати її

архітектуру, але й обрати ефективні методи оптимізації та регуляризації, які сприятимуть швидкому та стабільному збіженню.

Однією з ключових особливостей навчання гібридної моделі є вибір оптимального алгоритму оптимізації. У контексті нейронних мереж найбільш поширеними є такі алгоритми, як стохастичний градієнтний спуск (SGD), Adam та RMSprop. Стохастичний градієнтний спуск є базовим методом, який оновлює ваги моделі на основі градієнта функції втрат для кожного батчу даних. Проте цей метод може бути повільним та нестабільним при навчанні складних моделей з великою кількістю параметрів.

Алгоритм Adam є більш просунутим методом, який поєднує в собі переваги алгоритмів AdaGrad та RMSprop. Він адаптивно змінює швидкість навчання для кожного параметра, використовуючи оцінки першого та другого моментів градієнтів. Це дозволяє швидше досягати мінімуму функції втрат та забезпечує стабільніше навчання, особливо при роботі з великими та складними моделями. RMSprop також є адаптивним методом, який регулює швидкість навчання на основі середніх квадратів попередніх градієнтів, що допомагає уникати проблеми вибухаючих або затухаючих градієнтів.

Іншою важливою особливістю є застосування методів регуляризації для запобігання перенавчанню моделі. Перенавчання виникає, коли модель занадто добре пристосовується до навчальних даних і втрачає здатність узагальнювати на нових, невідомих даних. У випадку гібридних моделей з великою кількістю параметрів ця проблема є особливо актуальною. Техніки регуляризації, такі як dropout та L2-регуляризація, допомагають знизити ризик перенавчання. Dropout випадковим чином вимикає певний відсоток нейронів під час навчання, що запобігає надмірній залежності моделі від окремих нейронів та сприяє більш стійкому навчанню. L2-регуляризація додає штраф за великі значення ваг до функції втрат, стимулюючи модель до більш гладких та узагальнюючих рішень.

Важливим аспектом є також налаштування гіперпараметрів моделі, таких як швидкість навчання, розмір батчу, кількість шарів та кількість нейронів у кожному шарі. Ці параметри суттєво впливають на швидкість збіжності та

кінцеву точність моделі. Наприклад, занадто велика швидкість навчання може призвести до нестабільності та розбіжності, тоді як занадто мала — уповільнить процес навчання. Розмір батчу визначає кількість зразків, які обробляються одночасно, і впливає на обчислювальну ефективність та стабільність оцінки градієнтів.

При навчанні гібридної моделі особливу увагу слід приділити узгодженню роботи CNN та RNN компонентів. Конволюційні шари виділяють локальні ознаки зі спектрограми, тоді як рекурентні шари моделюють часові залежності між цими ознаками. Важливо забезпечити, щоб вихід CNN мав відповідний формат та розмір для введення в RNN. Це досягається шляхом правильного вибору розмірів фільтрів, кроків та паддінгу в конволюційних шарах.

Механізм уваги додає ще один рівень складності, оскільки вимагає обчислення ваг уваги для кожного часу та формування контекстного вектора. Це потребує додаткових обчислень та може збільшити час навчання. Однак механізм уваги суттєво покращує здатність моделі фокусуватися на найбільш релевантних частинах послідовності, що позитивно впливає на точність класифікації.

Під час навчання гібридної моделі може виникнути проблема вибухаючих або затухаючих градієнтів, особливо в рекурентних шарах. Це відбувається, коли значення градієнтів стають надто великими або надто малими, що заважає ефективному оновленню ваг. Для боротьби з цією проблемою застосовуються методи нормалізації градієнтів, такі як обрізання градієнтів (gradient clipping), де градієнти обмежуються певним максимальним значенням.

Ще однією особливістю є необхідність масштабування даних та використання пакетної нормалізації (batch normalization). Ця техніка нормалізує виходи шарів, що стабілізує навчання та дозволяє використовувати більші швидкості навчання. Це особливо корисно в глибоких мережах, де накопичення невеликих змін може призвести до значних відхилень на виході.

Важливим є також моніторинг процесу навчання за допомогою валідаційної вибірки. Це дозволяє відстежувати продуктивність моделі на даних,

які не були використані під час навчання, і вчасно виявляти ознаки перенавчання. Техніка ранньої зупинки (early stopping) може бути використана для зупинки навчання, коли точність на валідаційній вибірці перестає покращуватися, запобігаючи перенавчанню моделі.

Порівнюючи різні методи та підходи до навчання, можна зробити висновок, що вибір оптимізатора та технік регуляризації суттєво впливає на кінцеву продуктивність моделі. Адаптивні оптимізатори, такі як Adam, зазвичай забезпечують швидше та стабільніше навчання порівняно зі стохастичним градієнтним спуском. Застосування dropout та L2-регуляризації допомагає запобігти перенавчанню, хоча надмірна регуляризація може призвести до недонавчання моделі.

У підсумку, навчання гібридної моделі є багатогранним процесом, що вимагає збалансованого підходу до налаштування параметрів, вибору методів оптимізації та застосування технік регуляризації. Ретельний аналіз та експерименти з різними комбінаціями методів дозволяють знайти оптимальні налаштування, які забезпечать високу точність класифікації та здатність моделі узагальнювати на нових даних. Цей процес є критичним для успішного застосування гібридних моделей у задачах класифікації музики та інших складних обчислювальних завданнях.

2.6 Розробка архітектури вебсистеми

При проектуванні архітектури було прийнято рішення розділити систему на клієнтську та серверну частини, що взаємодіють між собою через стандартизовані протоколи зв'язку [15] (рис 2.2). Такий підхід забезпечує модульність, дозволяє незалежно розвивати та оновлювати кожен компонент системи, а також сприяє її гнучкості та масштабованості.

Серверна частина відповідає за обробку запитів від клієнтів, виконання класифікації музичних творів та управління даними. Вона реалізує програмний інтерфейс, який дозволяє приймати аудіофайли, виконувати необхідну попередню обробку, передавати дані в гібридну модель та повертати результати.

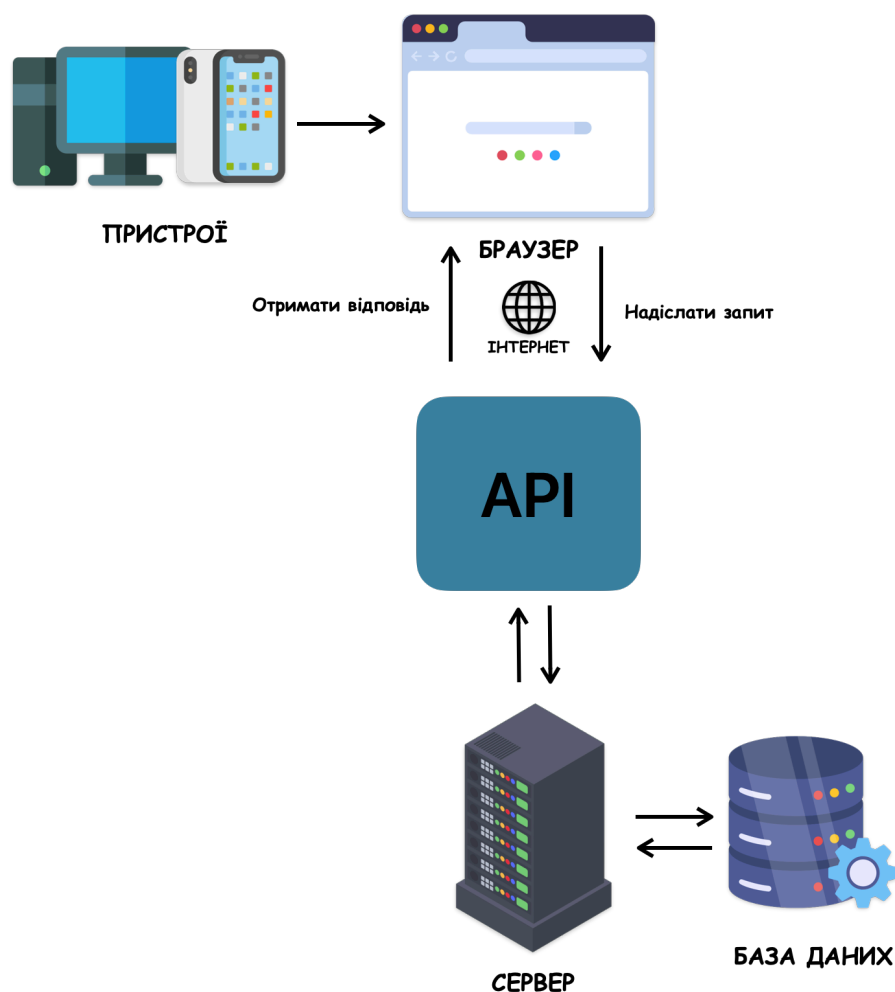


Рисунок 2.2 – Демонстрація клієнт-серверної взаємодії з використанням API

Клієнтська частина вебсистеми надає користувачам зручний інтерфейс для взаємодії з системою. Інтерфейс розроблено з урахуванням сучасних тенденцій дизайну та користувацького досвіду, що дозволяє легко завантажувати аудіофайли, переглядати результати класифікації та отримувати додаткову інформацію. Особлива увага приділяється адаптивності інтерфейсу, щоб забезпечити зручність користування на різних пристроях, включаючи мобільні телефони та планшети.

Взаємодія між клієнтською та серверною частинами здійснюється через стандартизовані протоколи, що забезпечує надійність та безпеку передачі даних. Такий підхід також спрощує інтеграцію з іншими сервісами та системами, дозволяючи розширювати функціональність платформи та підвищувати її цінність для користувачів.

Для зберігання даних використовується надійна система управління базами даних, яка забезпечує швидкий доступ до інформації та підтримує цілісність даних. Це дозволяє ефективно зберігати історію запитів, результати класифікації та іншу необхідну інформацію, а також забезпечує можливості для подальшого аналізу та вдосконалення системи.

Архітектура вебсистеми спроектована таким чином, щоб підтримувати масштабованість та високу доступність. Це означає, що при зростанні кількості користувачів або навантаження можна легко розширити ресурси системи без необхідності вносити значні зміни в код або інфраструктуру. Такий підхід забезпечує стабільну роботу сервісу та позитивний користувацький досвід навіть при високому навантаженні.

Особлива увага приділяється оптимізації продуктивності системи. Використовуються ефективні алгоритми та методи кешування для зниження часу відгуку та забезпечення швидкої обробки запитів. Це важливо для підтримки високого рівня задоволеності користувачів та конкурентоспроможності сервісу.

Моніторинг та логування є невід'ємною частиною архітектури вебсистеми. Вони дозволяють відстежувати роботу системи в реальному часі, виявляти та оперативно реагувати на можливі проблеми, аналізувати поведінку користувачів та планувати подальше вдосконалення платформи. Це сприяє підвищенню якості сервісу та забезпечує довіру користувачів.

При розробці архітектури було враховано принципи модульності та розширюваності. Кожен компонент системи виконує чітко визначені функції та може бути розроблений, протестований та розгорнутий окремо. Це спрощує процес розробки, дозволяє швидко впроваджувати нові функціональні можливості та адаптуватися до змінних вимог ринку.

Загалом, архітектура вебсистеми створена з акцентом на ефективність, безпеку та зручність для користувачів. Вона поєднує в собі передові технологічні рішення та орієнтована на надання високоякісного сервісу з класифікації музичних творів. Такий підхід забезпечує платформі конкурентні переваги та

створює міцну основу для подальшого розвитку та вдосконалення.

Діаграма послідовності для вебсистеми класифікації музичних творів на рисунку 2.3 детально демонструє порядок взаємодії між користувачем та основними компонентами системи під час процесу класифікації.

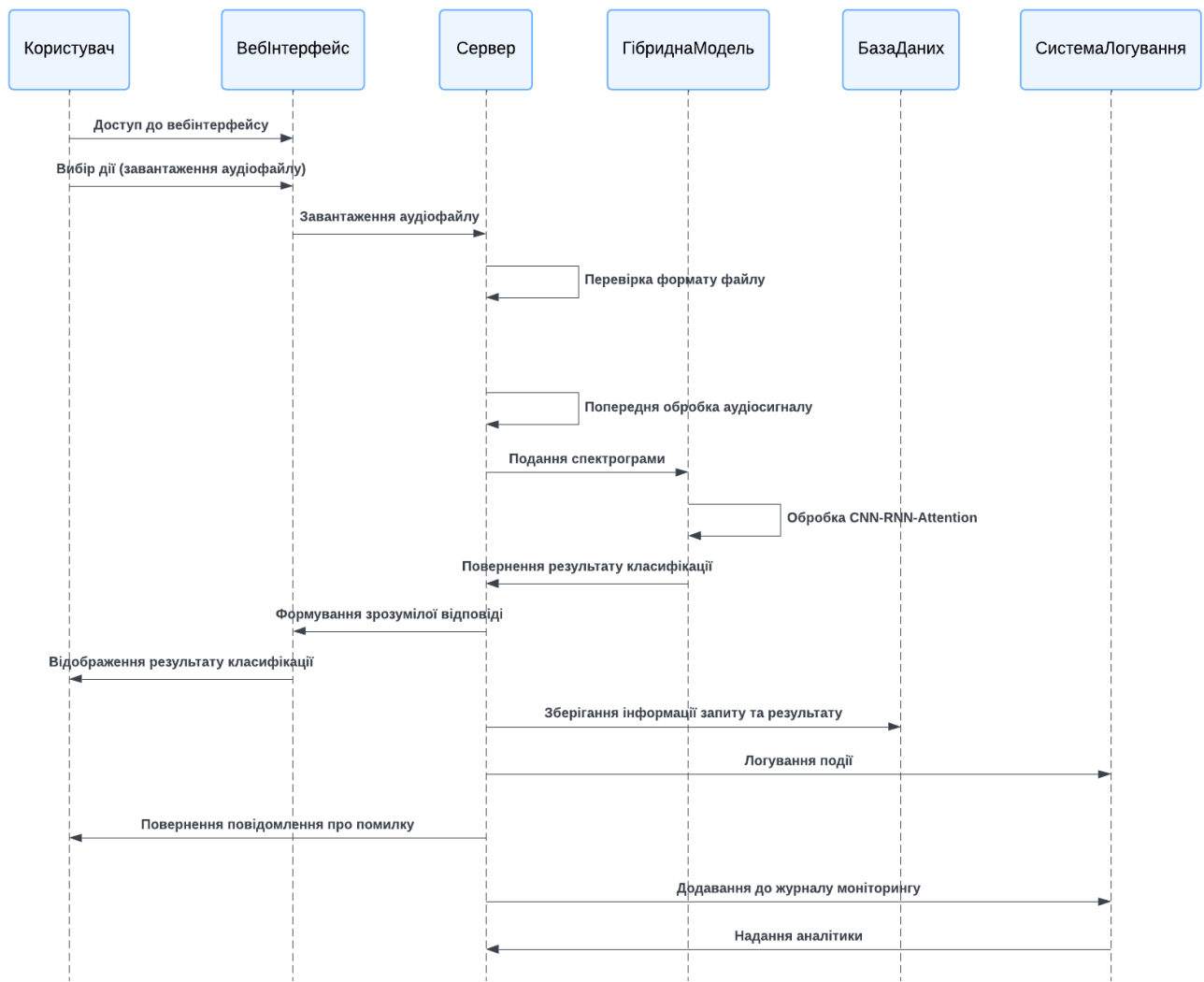


Рисунок 2.3 – Діаграма послідовності

Все починається з того, що користувач ініціює завантаження аудіофайлу через вебінтерфейс. Інтерфейс слугує точкою входу в систему, де користувач взаємодіє з вебдодатком, завантажуючи необхідні дані. Після цього вебінтерфейс передає запит разом з аудіофайлом на сервер для подальшої обробки за допомогою протоколу HTTP.

На сервері починається процес перевірки отриманого файлу. Сервер

звертається до модуля перевірки для аналізу формату аудіофайлу. Цей модуль перевіряє відповідність формату встановленим вимогам та повертає результат на сервер. Якщо формат файлу не відповідає вимогам, сервер негайно надсилає повідомлення про помилку назад на вебінтерфейс, де користувач бачить відповідне сповіщення.

У разі відповідності формату сервер передає аудіофайл модулю попередньої обробки, який виконує створення спектрограми за допомогою короткочасного перетворення Фур'є (STFT). Після завершення попередньої обробки спектрограма повертається на сервер, де відбувається наступний етап роботи.

Отримавши спектрограму, сервер передає її на вхід гібридної моделі, яка складається з конволюційної нейронної мережі (CNN), рекурентної нейронної мережі (RNN) та механізму уваги (Attention). Ця модель проводить аналіз спектрограми, виділяє важливі ознаки, враховує часові залежності та фокусується на найрелевантніших частинах аудіосигналу для точного визначення жанру. Після обробки гібридна модель передає результати класифікації назад на сервер.

Сервер отримує результати класифікації та зберігає їх разом з інформацією про запит у базі даних для можливого подальшого аналізу та звітності. Паралельно сервер передає дані до модуля логування для запису інформації про подію, що дозволяє підтримувати моніторинг системи та виявляти потенційні проблеми. Нарешті, сервер формує відповідь і надсилає її на вебінтерфейс. Користувач отримує результат класифікації, який відображається на екрані вебдодатка, та може ознайомитися з відповідною інформацією про музичний жанр.

2.7 Проектування бази даних користувачів

При проектуванні бази даних для зберігання інформації про користувачів і пов'язані з ними дані було враховано вимоги до безпеки, масштабованості, цілісності та зручності подальшого використання системи. Основна мета полягає

у створенні стрункої структури, яка дасть змогу ефективно опрацьовувати запити, зберігати історію класифікацій, а також забезпечить можливості для розширення функціоналу надалі.

В основу покладено реляційну модель даних, оскільки вона забезпечує високий рівень цілісності, а також дозволяє легко керувати транзакціями та оптимізувати запити. Під час проектування було визначено сутності, які відображають основні концепти системи: користувач, історію класифікацій та інші пов'язані дані (наприклад, індивідуальні налаштування).

На рисунку 2.4 наведено основні сутності та їхні атрибути, а також ключі та зв'язки між ними.

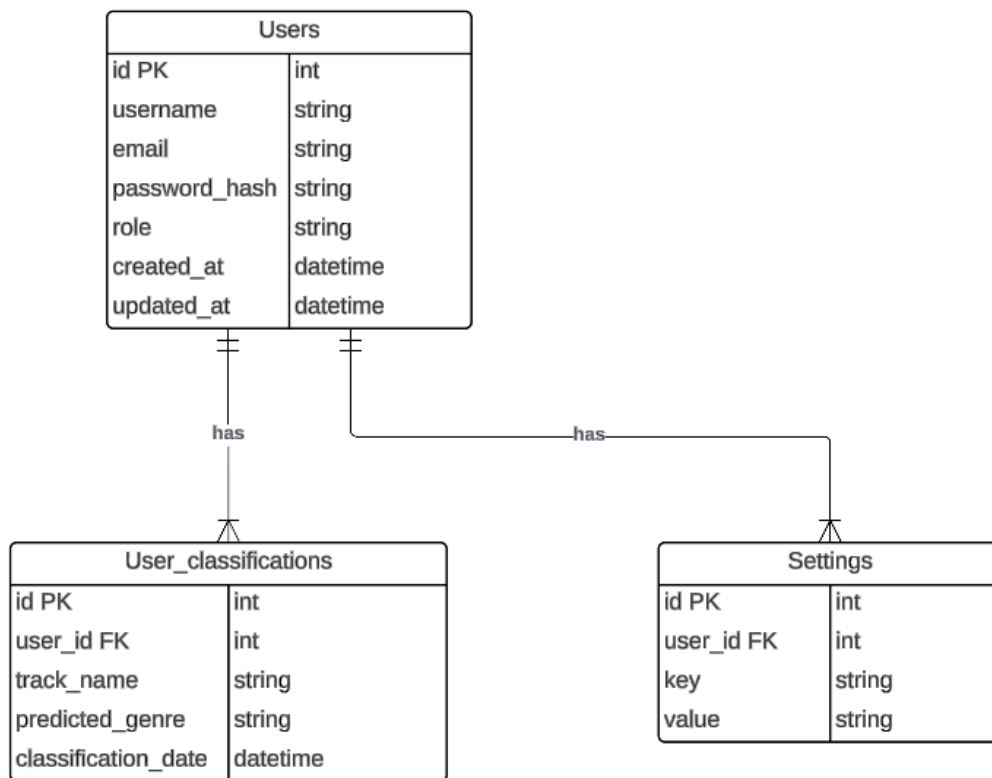


Рисунок 2.4 – Діаграма класів зв'язків бази даних

Users – сутність, яка зберігає інформацію про кожного зареєстрованого користувача. Атрибути: id (первинний ключ, унікальний ідентифікатор користувача), username (унікальне ім'я користувача для автентифікації), email (унікальна адреса електронної пошти), password_hash (хеш пароля для

безпечного зберігання), *role* (роль користувача, наприклад, «admin» або «user»), *created_at* (дата та час створення запису), *updated_at* (дата та час останнього оновлення даних).

User_classifications – сутність для зберігання історії класифікацій, виконаних користувачами, що дозволяє аналізувати їхню поведінку. Атрибути: *id* (первинний ключ, унікальний ідентифікатор класифікації), *user_id* (зовнішній ключ, пов'язаний із сутністю *Users*, визначає користувача, який виконав класифікацію), *track_name* (назва або ідентифікатор завантаженого треку), *predicted_genre* (визначений системою жанр), *classification_date* (дата та час виконання класифікації).

Settings (опційна сутність) – сутність для зберігання індивідуальних налаштувань користувачів. Атрибути: *id* (первинний ключ), *user_id* (зовнішній ключ, що зв'язує налаштування з конкретним користувачем), *key* (назва налаштування), *value* (значення налаштування).

2.8 Удосконалення процедури попередньої обробки аудіосигналу для гібридної моделі

У процесі вдосконалення процедури попередньої обробки аудіосигналу було впроваджено адаптивний підхід до вибору параметрів спектральних перетворень та методів нормалізації ознак з метою максимальної інформативності вхідних даних для гібридної моделі. Замість застосування фіксованих параметрів для всіх аудіозаписів, було розроблено алгоритмічний метод визначення оптимальних значень довжини вікна N , частоти дискретизації F_s , кроку перекриття R та типу віконної функції залежно від характеру музичного сигналу. Формально оптимізація здійснюється шляхом мінімізації функції втрат моделі за формулою:

$$N, F_s, R = \operatorname{argmin}_{N, F_s, R} \lambda(f(x; N; F_s, R)), \quad (2.14)$$

де:

x — вхідний аудіосигнал;

$f(x; N; F_s, R)$ — перетворення сигналу у спектрограму із заданими параметрами.

Крім адаптивного підбору параметрів, було впроваджено нормалізацію та стандартизацію спектральних коефіцієнтів (зокрема мел-кепстральних коефіцієнтів — MFCC), що дозволило зменшити залежність від гучності запису, фонових шумів та неконтрольованих амплітудних коливань. Якщо вхідні дані описуються вектором ознак x , операція нормалізації задається рівнянням:

$$\tilde{x} = \frac{x - \mu(x)}{\sigma(x)}, \quad (2.15)$$

де $\mu(x)$ та $\sigma(x)$ — середнє та стандартне відхилення елементів вектора ознак відповідно.

Завдяки цьому кожний набір ознак для різних записів приводиться до спільного масштабу, спрощуючи завдання моделі з їх аналізу.

Додаткове вдосконалення передбачало багаторівневий аналіз ознак: до базових спектральних коефіцієнтів додавалися їх часові похідні та прискорення, що дозволило відобразити динаміку змін ознак у часі. Наприклад, якщо m -й мел-кепстральний коефіцієнт у момент часу, то його перша та друга похідна обчислювалися за певним часовим вікном, відображаючи еволюцію спектральних характеристик. Це дозволило моделі враховувати не лише статичний тембр, а й ритмічні й мелодичні переходи, прискорення або уповільнення темпу, появу нових інструментів або вокалу.

У сукупності ці вдосконалення дали змогу підготувати більш репрезентативний набір вхідних даних для гібридної моделі, що поєднує CNN, RNN та механізм уваги. Адаптивний вибір параметрів спектрального аналізу, нормалізація ознак та багаторівневий аналіз їх змін у часі дозволили суттєво підвищити точність класифікації, прискорити процес обробки даних та

забезпечити більш гнучке підлаштування системи до різноманітних музичних жанрів і стильових особливостей. Це сприяло підвищенню інтерпретованості результатів та створенню більш стійкої до шумів та змінних умов експлуатації системи автоматичної класифікації музичних творів.

2.9 Висновки

У другому розділі було детально розглянуто архітектуру системи та реалізацію методів машинного навчання для класифікації музичних творів. Було проаналізовано особливості застосування конволюційних нейронних мереж (CNN), які виявилися надзвичайно ефективними для виділення локальних ознак у музичних сигналах, перетворених у двовимірні спектрограми.

Рекурентні нейронні мережі (RNN) також відіграють важливу роль, враховуючи часові залежності в музичних сигналах. Завдяки своїм зворотним зв'язкам RNN дозволяють моделі “пам'ятати” інформацію про попередні моменти часу, що є ключовим для аналізу музики як послідовності звуків. Використання методів оптимізації та регуляризації, таких як алгоритм Adam та техніки dropout, сприяє стабільному навчанню та запобігає перенавчанню моделі.

Впровадження механізму уваги (Attention) стало важливим доповненням, що дозволило покращити точність моделі за рахунок адаптивного фокусування на найбільш релевантних частинах аудіосигналу. Це забезпечує можливість моделі концентруватися на важливих фрагментах і підвищує її інтерпретованість. У результаті дослідження створено гібридну модель CNN-RNN-Attention, яка поєднує переваги кожного з методів та демонструє високу ефективність у класифікації музичних творів.

Особлива увага була приділена вдосконаленню процедури попередньої обробки аудіосигналу та екстракції релевантних ознак для гібридної моделі. Запроваджено адаптивний підхід до вибору параметрів спектральних перетворень, нормалізацію та стандартизацію коефіцієнтів, а також багаторівневий аналіз динаміки ознак у часі. Це дало змогу покращити

репрезентативність вхідних даних, підвищивши точність і швидкість обробки, а також зробило модель більш стійкою до змінних умов та різноманіття музичних жанрів.

Було розроблено архітектуру вебсистеми, яка забезпечує надійну обробку даних, масштабованість і високу продуктивність. Архітектура побудована за принципами модульності та розширюваності, що дозволяє легко інтегрувати нові функції та адаптувати систему під потреби користувачів. Розроблена вебсистема відкриває нові можливості для автоматизованої класифікації музики, що відповідає вимогам сучасної музичної індустрії та покращує досвід користувачів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ТА ЕКСПЕРИМЕНТАЛЬНИЙ АНАЛІЗ МОДЕЛІ КЛАСИФІКАЦІЇ МУЗИЧНИХ ТВОРІВ

3.1 Вибір мови програмування

У процесі розробки системи класифікації музичних творів за жанрами важливим етапом є вибір відповідних мов програмування та середовищ розробки, які забезпечать ефективність, продуктивність і зручність у роботі. Для нейронної мережі обрано Python з використанням PyCharm, а для веб-застосунку – React для фронтенду та Node.js для бекенду з використанням Visual Studio Code. Python є однією з найпопулярніших мов програмування для задач машинного навчання та обробки даних [16]. Основні причини вибору Python :

1. Широка екосистема бібліотек: Python має великий набір бібліотек для роботи з нейронними мережами, таких як TensorFlow, Keras, PyTorch, а також для обробки аудіо – LibROSA. Це значно полегшує процес розробки, тестування та оптимізації моделей.

2. Простота синтаксису: Python дозволяє швидко писати і тестувати код завдяки інтуїтивно зрозумілому синтаксису.

3. Активна спільнота: Python має велику кількість ресурсів, документації та прикладів коду.

4. Кросплатформеність: Python підтримується на всіх основних платформах, що дозволяє легко розгорнути нейронну мережу на різних серверах

Таблиця 3.1 – Порівняльна таблиця Python з іншими мовами

Мова	Бібліотеки для ML	Продуктивність	Простота синтаксису	Спільнота	Підтримка аудіообробки
Python	TensorFlow, Keras, PyTorch, LibROSA	Висока на CPU/GPU	Висока	Дуже велика	Висока
C#	ML.NET	Висока на .NET	Середня	Середня	Низька
Java	Deeplearning4j	Висока	Середня	Велика	Середня

Продовження таблиці 3.1

Мова	Бібліотеки для ML	Продуктивність	Простота синтаксису	Спільнота	Підтримка аудіообробки
R	TensorFlow, Keras	Низька	Середня	Середня	Низька
Julia	Flux.jl, Turing.jl	Дуже висока	Середня	Невелика	Низька

Node.js використовується для створення бекенд-системи, що буде обслуговувати запити, обробляти дані та взаємодіяти з нейронною мережею, написаною на Python [17]. Основні причини вибору Node.js для бекенду:

1. Асинхронна архітектура: Node.js використовує неблокуючу модель I/O, що робить його ідеальним для роботи з великими обсягами одночасних запитів.
2. Швидка обробка запитів: Завдяки своїй архітектурі Node.js є одним з найбільш швидких рушіїв для веб-серверів, що важливо для систем, які обробляють дані в реальному часі.
3. Легка інтеграція з фронтендом: Оскільки Node.js і фронтенд написані на JavaScript, це спрощує розробку і підтримку проєкту.

Таблиця 3.2 – Порівняльна таблиця Node.js з іншими рішеннями

Мова	Продуктивність	Легкість використання	Підтримка асинхронних запитів	Модульність	Інтеграція з фронтендом
Node.js	Висока	Висока	Так	Висока	Відмінна
Python (Flask)	Середня	Висока	Частково	Середня	Середня
Java	Висока	Середня	Так	Висока	Низька
Ruby	Середня	Висока	Ні	Середня	Середня

React був обраний для розробки інтерфейсу користувача завдяки своїй компонентній архітектурі та високій продуктивності. Основні переваги React:

1. Компонентний підхід: React дозволяє створювати повторно використовувані компоненти, що значно полегшує розробку та підтримку

великих проєктів.

2. Швидка робота з DOM: Використання віртуального DOM робить React дуже продуктивним при роботі з динамічними інтерфейсами.

3. Широка екосистема: React має велику кількість сторонніх бібліотек та плагінів, які полегшують інтеграцію нових функцій.

Таблиця 3.3 –Порівняльна таблиця React з іншими рішеннями

Фреймворк	Продуктивність	Простота використання	Спільнота	Масштабованість	Модульність
React	Висока	Висока	Дуже велика	Висока	Висока
Angular	Висока	Середня	Велика	Висока	Висока
Vue.js	Висока	Висока	Велика	Середня	Висока
Svelte	Дуже висока	Висока	Зростає	Середня	Висока

3.2 Вибір середовища програмування

PyCharm є одним із найкращих середовищ розробки для Python, особливо для проєктів у сфері машинного навчання [18]. Основні переваги PyCharm:

1. Інтеграція з бібліотеками: PyCharm легко інтегрується з такими бібліотеками, як TensorFlow, Keras і PyTorch.

2. Інструменти для налагодження та тестування: IDE надає потужні інструменти для тестування, налагодження та інтеграції з науковими інструментами.

3. Інтеграція з Git та іншими системами контролю версій: PyCharm дозволяє легко працювати з репозиторіями, що важливо при командній роботі.

Таблиця 3.4 – Порівняльна таблиця PyCharm з іншими середовищами

IDE	Підтримка Python бібліотек	Інструменти налагодження	Інтеграція з Git	Інтерфейс	Кросплатформеність
PyCharm	Відмінна	Відмінна	Так	Дуже зручний	Висока

Продовження таблиці 3.4

IDE	Підтримка Python бібліотек	Інструменти налагодження	Інтеграція з Git	Інтерфейс	Кросплатформеність
Visual Studio Code	Дуже хороша	Середня	Так	Зручний	Висока
Jupyter Notebook	Середня	Немає	Так	Дуже зручний	Висока
Spyder	Хороша	Середня	Немає	Середній	Висока

Visual Studio Code (VSCode) є найпопулярнішим редактором коду для роботи з JavaScript та Node.js завдяки своїм численним розширенням і зручному інтерфейсу. Основні переваги VSCode:

1. Широка підтримка плагінів: VSCode підтримує безліч плагінів для роботи з React та Node.js, які забезпечують автодоповнення, аналіз коду та інтеграцію з системами контролю версій.
2. Інтеграція з Git: VSCode надає потужні інструменти для роботи з Git прямо з редактора, що полегшує командну розробку.
3. Вбудовані інструменти для налагодження: VSCode дозволяє налагоджувати код для Node.js і React безпосередньо в редакторі.

Таблиця 3.5 – Порівняльна таблиця VSCode з іншими середовищами

Редактор	Автодоповнення коду	Інтеграція з Git	Підтримка розширень	Налагодження	Модульність
VSCode	Відмінна	Так	Дуже велика	Вбудоване	Висока
WebStorm	Відмінна	Так	Велика	Вбудоване	Висока
Sublime Text	Хороша	Частково	Середня	Немає	Висока
Svelte	Дуже висока	Висока	Зростає	Середня	Висока

VSCode є чудовим вибором для розробки як фронтенду (React), так і бекенду (Node.js) завдяки своїм потужним інструментам та інтеграції з системами контролю версій.

Отже, Visual Studio Code (VSCode) є ідеальним середовищем для розробки як фронтенду на React, так і бекенду на Node.js завдяки своїм потужним інструментам. Його підтримка плагінів робить роботу з JavaScript і Node.js ефективною та зручною. Python обрано для нейронної мережі через його багату екосистему бібліотек, таких як TensorFlow, Keras, та PyTorch, що дозволяє швидко й ефективно працювати з моделями машинного навчання. Використання PyCharm забезпечує зручне налагодження та ефективну роботу з даними. Node.js використовується для серверної частини завдяки своїй асинхронній архітектурі, яка дозволяє обробляти велику кількість одночасних запитів. Використання Node.js разом із React забезпечує тісну інтеграцію між фронтом і бекендом, що спрощує розробку через єдину мову програмування – JavaScript. Таким чином, поєднання Python для нейронних мереж та Node.js з React для веб-застосунку в середовищах PyCharm і VSCode створює потужний та ефективний стек технологій для розробки системи класифікації музичних творів.

3.3 Програмна реалізація конволюційних нейронних мереж (CNN) для класифікації музики

Після аналізу в розділі 2.1, необхідно реалізувати конволюційні нейронні мережі у коді. CNN є основним інструментом для класифікації музичних творів, адже вони здатні виділяти ключові ознаки з даних у форматі спектрограм.

Спектрограми є двовимірним представленням аудіосигналу, де одна вісь відображає час, а інша частоту, а значення пікселів відповідають амплітудам сигналу. Для отримання спектрограм [19] використовують короткочасне перетворення Фур'є (STFT), яке розкладає сигнал на частотні компоненти.

```
import numpy as np  
from scipy.signal import stft
```

```
def compute_stft(audio_signal, sample_rate, window='hann', nperseg=256):
    f, t, Zxx = stft(audio_signal, fs=sample_rate, window=window,
nperseg=nperseg)
    return f, t, np.abs(Zxx)
```

Використання STFT дозволяє створити двовимірне зображення – спектрограму, яка зручна для аналізу за допомогою CNN. Після отримання спектрограми проводиться її візуалізація та подальша обробка.

```
sample_rate = 44100 # Частота дискретизації
signal = np.sin(2 * np.pi * 440 * np.linspace(0, 5, sample_rate * 5))
frequencies, times, spectrogram = compute_stft(signal, sample_rate)
```

Конволюційний шар виконує операцію згортки для виділення локальних ознак у спектрограмі, таких як частотні патерни, що характерні для певних музичних жанрів.

```
def convolution2d(input_matrix, kernel):
    output_height = input_matrix.shape[0] - kernel.shape[0] + 1
    output_width = input_matrix.shape[1] - kernel.shape[1] + 1
    output_matrix = np.zeros((output_height, output_width))
```

Цей процес дозволяє згорнути спектрограму з різними фільтрами, виділяючи важливі елементи.

```
kernel = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
output_matrix = convolution2d(spectrogram, kernel)
```

Результати згортки передаються через функцію активації для додання нелінійності. Це допомагає моделі краще моделювати складні залежності.

```
def relu(x):
    return np.maximum(0, x)
activated_output = relu(output_matrix)
```

Функція ReLU сприяє тому, що модель акцентує увагу на позитивних значеннях, ігноруючи негативні

Для зменшення розмірності даних застосовують Max Pooling, який залишає найважливіші ознаки, знижуючи при цьому обчислювальні витрати.

```
def max_pooling2d(input_matrix, pool_size):
    output_height = input_matrix.shape[0] // pool_size
    output_width = input_matrix.shape[1] // pool_size
    pooled_matrix = np.zeros((output_height, output_width))
```

Ця операція дозволяє скоротити кількість оброблюваних даних, зберігаючи при цьому ключову інформацію.

```
pooled_output = max_pooling2d(activated_output, 2)
```

Після згорткових і пулінгових шарів дані розгортаються для передачі до повнозв'язаних шарів, де проводиться остаточна класифікація.

```
def fully_connected(input_vector, weights, biases):
    return np.dot(input_vector, weights) + biases
```

Повнозв'язані шари поєднують виділені ознаки для створення прогнозу ймовірностей класів.

```
input_vector = pooled_output.flatten()
weights = np.random.rand(len(input_vector), 10)
```

```
biases = np.random.rand(10)
output_vector = fully_connected(input_vector, weights, biases)
```

Для остаточного рішення застосовується функція softmax, яка нормалізує значення до ймовірностей.

```
def softmax(x):
    exp_x = np.exp(x - np.max(x))
    return exp_x / exp_x.sum(axis=0)
probabilities = softmax(output_vector)
```

Softmax перетворює вихід моделі в набір значень, що представляють ймовірність приналежності до різних жанрів музики. Таким чином, за допомогою CNN можна ефективно класифікувати музичні твори, аналізуючи складні патерни у спектрограмі та використовуючи їх для передбачення жанру.

3.4 Програмна реалізація рекурентних нейронних мереж (RNN) для класифікації музики

Після реалізації конволюційної нейронної мережі (CNN), ми переходимо до рекурентної нейронної мережі (RNN), яка дозволяє враховувати часові залежності в музичному сигналі. Використовуючи дані, отримані з CNN, ми передаємо їх до RNN для подальшого аналізу.

Спочатку перетворюємо вихід CNN у послідовність, придатну для введення в RNN. Це досягається шляхом зміни розмірності вихідного тензора CNN:

```
from tensorflow.keras.layers import Reshape
cnn_output = cnn_model.output # Вихід з CNN
sequence_length = cnn_output.shape[1]
feature_dim = cnn_output.shape[2] * cnn_output.shape[3]
sequence_input = Reshape((sequence_length, feature_dim))(cnn_output)
```

Тут ми використовуємо шар `Reshape`, щоб перетворити тривимірний тензор ознак у двовимірну послідовність, де кожен крок відповідає певному моменту часу.

Далі додаємо рекурентний шар, наприклад, LSTM, для обробки послідовності ознак та врахування часових залежностей:

```
from tensorflow.keras.layers import LSTM  
rnn_output = LSTM(128, activation='tanh')(sequence_input)
```

Шар `LSTM` з 128 нейронами обробляє послідовність та генерує вихідний вектор, який містить інформацію про всю послідовність вхідних даних.

Після рекурентного шару додаємо повнозв'язаний шар для класифікації на основі виходу RNN:

```
from tensorflow.keras.layers import Dense  
output = Dense(num_classes, activation='softmax')(rnn_output)
```

Цей шар перетворює вихід RNN у вектор ймовірностей приналежності до кожного з музичних жанрів, використовуючи функцію активації `softmax`.

Об'єднуємо CNN та RNN в одну модель для кінцевої класифікації:

```
from tensorflow.keras.models import Model  
model = Model(inputs=cnn_model.input, outputs=output)
```

Таким чином, ми отримуємо модель, яка інтегрує CNN та RNN, що дозволяє враховувати як просторові, так і часові ознаки музичного сигналу.

У результаті, наша модель готова до подальшого використання в механізмі уваги (Attention), який допоможе фокусуватися на найбільш важливих частинах послідовності та покращити точність класифікації.

3.5 Програмна реалізація механізму уваги (Attention) у класифікації музичних творів

Після інтеграції конволюційних (CNN) та рекурентних (RNN) нейронних мереж, наступним кроком є впровадження механізму уваги (Attention) для покращення точності класифікації музичних творів. Механізм уваги дозволяє моделі фокусуватися на найбільш релевантних частинах послідовності, що є особливо важливим для музичних даних, де ключові особливості можуть бути розподілені нерівномірно в часі.

Почнемо з додавання шару уваги [20] після RNN. Вихід RNN складається з послідовності прихованих станів, які можна використовувати для обчислення ваг уваги. Спочатку визначимо функцію оцінки важливості для кожного тимчасового кроку:

```
import tensorflow as tf
from tensorflow.keras.layers import Dense
def attention_scores(hidden_states):
    score_first_part = Dense(64, activation='tanh')(hidden_states)
```

У цій функції ми застосовуємо повнозв'язаний шар з активацією `tanh` для обчислення проміжних оцінок. Далі обчислюємо ваги уваги за допомогою функції softmax, щоб отримати ймовірності:

```
attention_weights = tf.nn.softmax(score_first_part, axis=1)
context_vector = attention_weights * hidden_states
```

Контекстний вектор отримуємо шляхом зваженого множення ваг уваги на відповідні приховані стани. Потім сумуємо ці значення по часовій осі:

```
context_vector = tf.reduce_sum(context_vector, axis=1)
return context_vector
```

Цей контекстний вектор містить агреговану інформацію про найбільш важливі частини послідовності. Інтегруємо механізм уваги в модель, додавши його після шару RNN:

```
rnn_output = lstm_layer(sequence_input)
context_vector = attention_scores(rnn_output)
```

Після отримання контекстного вектора додаємо вихідний шар для класифікації:

```
output = Dense(num_classes, activation='softmax')(context_vector)
```

Компонуємо повну модель, яка поєднує CNN, RNN та механізм уваги:

```
from tensorflow.keras.models import Model
model = Model(inputs=cnn_model.input, outputs=output)
```

Компілюємо модель з оптимізатором `'adam'` та функцією втрат `'categorical_crossentropy'` для багатокласової класифікації:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['ac  
curacy'])
```

Тепер модель готова до навчання на даних музичних творів. Використовуючи механізм уваги, модель зможе ефективніше фокусуватися на ключових моментах в аудіосигналі, що підвищить точність класифікації за жанрами.

Таким чином, ми успішно реалізували механізм уваги в нашій моделі, що поєднує в собі CNN для просторових ознак, RNN для часових залежностей та Attention для виділення найбільш релевантних частин послідовності. Це

завершує етап підготовки моделі до навчання та подальшого використання у класифікації музичних творів.

3.6 Реалізація клієнт-сервер архітектури

У процесі розробки вебдодатку важливим завданням було інтегрувати навчений алгоритм на основі CNN-RNN-Attention у серверну частину. Це дозволяє виконувати класифікацію музичних творів у режимі реального часу, забезпечуючи швидку та точну відповідь для користувача. Серверна частина побудована на базі фреймворку Flask, який надає гнучкий інструментарій для створення вебзастосунків на Python.

Після навчання модель зберігається у файл для подальшого використання. При запуску сервера модель завантажується:

```
import tensorflow as tf
model = tf.keras.models.load_model('saved_model.h5')
```

У Flask-додатку визначається маршрут, який обробляє запити від клієнта:

```
from flask import Flask, request, jsonify
app = Flask(__name__)
```

У функції `classify` отримуємо аудіофайл з запиту:

```
file = request.files['audio']
y, sr = librosa.load(file, sr=22050)
```

Перетворюємо аудіофайл у спектрограму та готуємо дані для моделі:

```
spectrogram = librosa.feature.melspectrogram(y=y, sr=sr)
spectrogram_db = librosa.power_to_db(spectrogram, ref=np.max)
```

```
spectrogram_db = np.expand_dims(spectrogram_db, axis=(0, -1))
```

Виконуємо передбачення жанру за допомогою моделі:

```
prediction = model.predict(spectrogram_db)
genre_index = np.argmax(prediction, axis=1)[0]
```

Створюємо список жанрів і отримуємо назву жанру та повертаємо результат у форматі json:

```
genres = ['Блюз', 'Класика', 'Кантрі', 'Диско',
          'Хіп-хоп', 'Джаз', 'Метал', 'Поп', 'Реггі', 'Рок']
genre = genres[genre_index]
return jsonify({'genre': genre})
```

Цей код забезпечує інтеграцію моделі в серверну частину. Коли користувач завантажує аудіофайл через вебінтерфейс, сервер обробляє запит і повертає результат класифікації.

На клієнтській стороні використовується JavaScript для відправки файлу та отримання результату.

Для збереження інформації про запити та результати класифікації інтегровано базу даних у серверну частину. Використовується база даних SQLite з бібліотекою SQLAlchemy [22] для зручної роботи з даними.

Інтеграція моделі CNN-RNN-Attention у серверну частину здійснена шляхом завантаження навченої моделі при старті сервера та використання її для передбачення жанру після попередньої обробки аудіофайлів. Логування результатів класифікації до бази даних та журналу підвищує надійність системи і дозволяє аналізувати її роботу.

Використання Flask та SQLAlchemy спрощує роботу з базою даних і забезпечує ефективну обробку запитів. Користувачі можуть швидко отримувати

результати класифікації, а розробники мають можливість покращувати систему на основі зібраних даних. Логування до бази даних дозволяє зберігати історію класифікацій, що корисно для статистичного аналізу та покращення моделі. Це відкриває можливості для розширення функціональності системи, наприклад, додавання персоналізованих рекомендацій на основі попередніх запитів.

3.7 Програмна реалізація удосконалення процедури попередньої обробки аудіосигналу

Подальший розвиток методу попередньої обробки аудіосигналу було досягнуто завдяки впровадженню алгоритмів адаптивного вибору спектрально-часових характеристик та фокусуванню уваги на ключових фрагментах сигналу. Цей підхід значно відрізняється від класичних методів, оскільки дозволяє ефективніше виділяти релевантні ознаки, що впливають на точність класифікації. Програмна реалізація цих удосконалень включає нові алгоритми для автоматизованого аналізу структури сигналу та оптимізації обробки даних.

Основою адаптивного вибору спектрально-часових характеристик є динамічний аналіз параметрів сигналу, таких як довжина вікна, крок перекриття та розподіл енергії. Наприклад, для кожного аудіосигналу спочатку обчислюється енергетичний профіль, що дозволяє ідентифікувати ключові фрагменти із найвищою інформативністю. Такий підхід значно зменшує обсяг даних, які потребують обробки, і фокусується на найбільш важливих частинах сигналу.

У коді це реалізується через функцію, яка визначає фрагменти із високим енергетичним значенням:

```
import numpy as np

def extract_key_segments(signal, segment_length, overlap):
    energy = np.array([np.sum(signal[i:i + segment_length] ** 2)
                     for i in range(0, len(signal) - segment_length, overlap)])
    threshold = np.mean(energy) + np.std(energy)
```

```
key_segments = [i for i, e in enumerate(energy) if e > threshold]
return key_segments
```

У цьому фрагменті, «energy» обчислює суму квадратів амплітуд сигналу у кожному сегменті, що відповідає його енергії, «threshold» визначає порогове значення для виявлення інформативних сегментів, що базується на середньому і стандартному відхиленні енергії, «key_segments» повертає список індексів ключових фрагментів.

Наступним кроком є витягнення ознак із вибраних ключових сегментів. Для цього до сигналу застосовуються спектральні перетворення тільки у межах виявлених сегментів.

```
def extract_features_from_segments(signal, sr, key_segments, segment_length):
    features = []
    for seg in key_segments:
        start = seg * segment_length
        end = start + segment_length
        segment = signal[start:end]
        mfcc = librosa.feature.mfcc(y=segment, sr=sr, n_mfcc=13)
        features.append(np.mean(mfcc, axis=1))
    return np.array(features)
```

У цьому фрагменті «key_segments» задає індекси сегментів із найбільшою інформативністю, «librosa.feature.mfcc()» обчислює MFCC, які використовуються як базові ознаки, «np.mean()» агрегує ознаки по всьому сегменту, створюючи узагальнений вектор ознак.

Для фокусування уваги на ключових фрагментах застосовано підхід, що базується на вагових коефіцієнтах. У результаті кожному фрагменту сигналу присвоюється ваговий коефіцієнт, який відображає його значущість. Це дозволяє моделі приділяти більше уваги релевантним частинам сигналу під час навчання.

```
def calculate_attention_weights(energy):
    total_energy = np.sum(energy)
    weights = energy / total_energy
    return weights
```

Де, «energy» — масив енергетичних значень для кожного сегменту, «weights» — нормалізовані вагові коефіцієнти, які використовуються під час навчання моделі.

Такий підхід до попередньої обробки сигналів дозволив значно підвищити достовірність класифікації завдяки адаптивному вибору спектрально-часових характеристик. Крім того, зменшення кількості оброблюваних даних скоротило час обчислень, забезпечивши ефективність та оптимальність роботи системи.

3.8 Програмна реалізація бази даних клієнта

Для програмної реалізації бази даних клієнтів у системі класифікації було створено набір класів у Python, які представляють таблиці бази даних. Кожен із цих класів включає атрибути, що відповідають колонкам у таблицях, а також методи для взаємодії з даними. Код реалізації базується на бібліотеці SQLAlchemy, яка забезпечує зручну і гнучку інтеграцію між об'єктами Python і реляційною базою даних.

Початкова конфігурація бази даних включає створення базового класу за допомогою `declarative\_base`, що слугує основою для всіх таблиць:

```
from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()
```

Таблиця користувачів, наприклад, представлена класом `User`. Кожна властивість класу (`id`, `username`, `email`, тощо) визначається як колонка у таблиці бази даних. Колонки створюються за допомогою типів SQLAlchemy, таких як `Integer`, `String`, `DateTime`. Наприклад, поле `id` визначається як первинний ключ із використанням параметра `primary\_key=True`, що гарантує

унікальність кожного запису.

```
from sqlalchemy import Column, Integer, String, DateTime
from datetime import datetime

class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    username = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    password_hash = Column(String, nullable=False)
    role = Column(String, default='user')
    created_at = Column(DateTime, default=datetime.utcnow)
    updated_at = Column(DateTime, default=datetime.utcnow)
```

Для зберігання результатів класифікації використовується клас `UserClassification`, який включає зовнішній ключ, що зв'язує цей запис із відповідним користувачем у таблиці `User`. Додатково визначено поля для збереження назви треку, передбаченого жанру та дати класифікації. Кожне поле створюється окремо, а зовнішній ключ реалізується через параметр `ForeignKey`.

```
from sqlalchemy import ForeignKey

class UserClassification(Base):
    __tablename__ = 'user_classifications'
    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('users.id'))
    track_name = Column(String, nullable=False)
    predicted_genre = Column(String, nullable=False)
    classification_date = Column(DateTime, default=datetime.utcnow)
```

Щоб створити базу даних, використовується метод `create\_all` бібліотеки

SQLAlchemy. Цей метод перевіряє, чи існують таблиці в базі даних, і, якщо ні, автоматично їх створює. Це дозволяє уникнути дублювання таблиць або помилок під час їх ініціалізації.

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///music_classification.db')
Base.metadata.create_all(engine)
```

Для роботи з даними у базі (наприклад, додавання нового користувача) використовується сесія SQLAlchemy, створена через `sessionmaker`. Сесія дозволяє виконувати транзакції з базою даних. Для додавання нового користувача спочатку створюється об'єкт класу `User`, після чого він додається до сесії і комітиться для збереження змін у базі.

```
from sqlalchemy.orm import sessionmaker

Session = sessionmaker(bind=engine)
session = Session()

def add_user(username, email, password_hash):
    new_user = User(username=username, email=email,
password_hash=password_hash)
    session.add(new_user)
    session.commit()
```

Крім додавання, реалізовано також методи оновлення даних. Наприклад, щоб змінити адресу електронної пошти користувача, у таблиці знаходиться відповідний запис за його ідентифікатором (`id`), після чого значення поля `email` змінюється. Зміни зберігаються через метод `commit`.

```
def update_user_email(user_id, new_email):
```

```

user = session.query(User).filter(User.id == user_id).first()
if user:
    user.email = new_email
    user.updated_at = datetime.utcnow()
    session.commit()

```

Таким чином, вся структура бази даних реалізована у вигляді компактного, але потужного набору класів Python. SQLAlchemy забезпечує не лише автоматизацію створення таблиць, але й простий доступ до даних, що значно полегшує процес розробки.

3.9 Висновки

У цьому розділі було детально описано процес розробки вебдодатку для автоматичної класифікації музичних творів за жанрами. Вибір мов програмування та середовищ розробки був обґрунтований з точки зору ефективності, продуктивності та зручності використання. Для реалізації нейронної мережі було обрано Python з використанням PyCharm, що дозволило ефективно працювати з моделями машинного навчання завдяки широкій екосистемі бібліотек, таких як TensorFlow, Keras та LibROSA.

Для вебзастосунку були обрані React для фронтенду та Node.js для бекенду з використанням Visual Studio Code. Це забезпечило тісну інтеграцію між клієнтською та серверною частинами через єдину мову програмування JavaScript, що спростило процес розробки та підтримки системи.

Було реалізовано конволюційну нейронну мережу (CNN) для виділення просторових ознак зі спектрограм музичних творів. Додавання рекурентної нейронної мережі (RNN) дозволило врахувати часові залежності в аудіосигналі, а впровадження механізму уваги (Attention) підвищило точність класифікації, фокусуючись на найбільш релевантних частинах послідовності.

Було реалізовано покращений підхід до попередньої обробки аудіосигналів, що включає адаптивний вибір спектрально-часових

характеристик і фокусування уваги на ключових фрагментах сигналу, що дозволило значно підвищити якість класифікації за рахунок акценту на найбільш інформативних частинах сигналу та зменшення обчислювальної складності.

Реалізація клієнт-серверної архітектури з використанням Flask забезпечила інтеграцію навченої моделі в серверну частину, дозволяючи виконувати класифікацію в режимі реального часу. Логування результатів класифікації до бази даних SQLite з використанням SQLAlchemy підвищило надійність системи та надало можливість аналізувати її роботу для подальшого вдосконалення.

Загалом, розроблений вебдодаток поєднує сучасні технології та підходи, забезпечуючи ефективну та точну класифікацію музичних творів за жанрами. Система є гнучкою та масштабованою, що дозволяє легко адаптувати її до майбутніх вимог та розширювати функціональність відповідно до потреб користувачів і ринку. Це створює міцну основу для подальшого розвитку платформи та її успішного застосування в музичній індустрії.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

4.1 Аналіз методів тестування

У сучасному світі інформаційних технологій якість програмного забезпечення є одним із найважливіших аспектів його успішного впровадження та використання. Особливо це стосується систем, які працюють із великими обсягами даних та застосовують складні алгоритми машинного навчання, як-от розроблена система автоматичної класифікації музичних творів за жанрами.

Тестування програмного забезпечення [21] є невід'ємною складовою процесу розробки, яка забезпечує якість, надійність та відповідність продукту вимогам користувачів. Існує ряд методів тестування, кожен з яких має свою мету та застосовується в залежності від специфіки проекту та його стадії розробки.

Функціональне тестування спрямоване на перевірку того, чи відповідає програмне забезпечення своїм функціональним вимогам і специфікаціям. Воно передбачає перевірку всіх функцій та можливостей системи, включаючи обробку вхідних даних, коректність виводу результатів, роботу з користувацьким інтерфейсом та реакцію на дії користувача. Метою функціонального тестування є виявлення невідповідностей між очікуваною та реальною поведінкою системи, що дозволяє виправити помилки та забезпечити відповідність продукту вимогам.

Модульне тестування полягає в перевірці окремих компонентів або модулів системи в ізоляції від інших частин. Кожен модуль тестується окремо, що дозволяє виявити та виправити помилки на ранніх етапах розробки, коли їх усунення є менш витратним. Модульне тестування допомагає забезпечити правильну роботу кожного компонента, перш ніж вони будуть інтегровані в єдину систему.

Інтеграційне тестування проводиться після модульного тестування і спрямоване на перевірку взаємодії між різними модулями системи. Воно дозволяє виявити проблеми, які можуть виникнути при об'єднанні окремих компонентів, такі як невідповідність інтерфейсів, некоректна передача даних або конфлікти між модулями. Інтеграційне тестування забезпечує плавну взаємодію

всіх частин системи та її коректну роботу в цілому.

Системне тестування охоплює перевірку всієї системи як єдиного цілого в умовах, максимально наближених до реальних. Метою цього тестування є оцінка поведінки системи під різними умовами, включаючи навантаження, стресові ситуації та некоректні дії користувачів. Системне тестування дозволяє переконатися, що продукт відповідає всім вимогам та готовий до випуску.

Регресійне тестування використовується для виявлення дефектів, які можуть виникнути після внесення змін у код, оновлень або виправлень. Воно спрямоване на перевірку того, що нові зміни не негативно вплинули на вже існуючу функціональність системи. Це особливо важливо в умовах, коли продукт постійно розвивається та доповнюється новими можливостями.

Навантажувальне тестування спрямоване на оцінку продуктивності системи під різними рівнями навантаження. Воно допомагає визначити, як система поводить себе при великій кількості одночасних користувачів, великому обсязі даних або інтенсивних операціях. Навантажувальне тестування дозволяє виявити вузькі місця в продуктивності та оптимізувати систему для кращої роботи під навантаженням.

Тестування безпеки передбачає перевірку системи на наявність вразливостей та забезпечення конфіденційності, цілісності та доступності даних. Воно включає оцінку захисту від несанкціонованого доступу, атак типу SQL-ін'єкцій, міжсайтового скриптингу та інших потенційних загроз. Тестування безпеки є критично важливим для захисту даних користувачів та репутації продукту.

Для розробленої системи автоматичної класифікації музичних творів за жанрами було обрано застосування таких методів тестування: функціональне, модульне, інтеграційне та тестування алгоритмів машинного навчання.

Функціональне тестування було вибрано тому, що головною метою є перевірка відповідності системи її функціональним вимогам. Необхідно було переконатися, що система правильно класифікує музичні твори за жанрами, інтерфейс користувача працює коректно, а всі функції доступні та виконуються

належним чином.

Модульне тестування буде застосоване для перевірки окремих компонентів системи, таких як серверна частина та база даних. Це дозволить виявити та виправити помилки на ранніх стадіях розробки, забезпечивши коректну роботу кожного модуля до їх інтеграції.

Інтеграційне тестування буде необхідне для перевірки взаємодії між різними модулями системи. Оскільки система має клієнт-серверну архітектуру, важливо було переконатися в коректності передачі даних між клієнтом і сервером, взаємодії з базою даних та сумісності всіх компонентів.

Тестування алгоритмів машинного навчання є специфічним для систем, що використовують штучний інтелект. Воно було обрано для оцінки якості класифікації музичних творів, використовуючи метрики точності, повноти, точності передбачення та F1-міри. Це дозволить оцінити ефективність моделі, її здатність до узагальнення та виявити можливі упередження або помилки в класифікації.

Вибір саме цих методів тестування зумовлений специфікою розроблюваної системи та необхідністю забезпечити її високу якість, надійність та відповідність вимогам користувачів. Застосування комплексного підходу до тестування дозволило виявити та усунути потенційні дефекти, оптимізувати роботу системи та підготувати її до успішного впровадження.

4.2 Тестування функціональних можливостей програмного продукту

Тестування функціональних можливостей є важливим етапом у забезпеченні якості програмного продукту, оскільки воно підтверджує відповідність системи заявленим вимогам та очікуванням користувачів. У цьому розділі розглядається процес функціонального тестування розробленої системи автоматичної класифікації музичних творів за жанрами, описуються результати тестування та аналізуються виявлені недоліки.

Метою функціонального тестування було перевірити коректність класифікації музичних творів за жанрами, забезпечити безперебійну роботу

клієнтського інтерфейсу та його відповідність вимогам користувачів. Також важливо було переконатися у надійній взаємодії клієнтської та серверної частин системи, коректній обробці різних форматів аудіофайлів та їх розмірів, а також у стабільній роботі системи за різних умов використання, включаючи некоректні дії користувачів.

Для проведення функціонального тестування [22] було розроблено набір тестових сценаріїв, які охоплювали основні функціональні можливості системи. Кожен сценарій містив опис тесту, вхідні дані, очікуваний та фактичний результати. Тестування проводилося вручну на різних пристроях та веббраузерах, щоб забезпечити максимальне покриття можливих сценаріїв використання.

Основні тестові сценарії наведено в таблиці 4.1.

Таблиця 4.1 – Основні тестові сценарії функціонального тестування

Номер	Опис тестового сценарію	Вхідні дані	Очікуваний результат	Фактичний результат
1	Завантаження аудіофайлу коректного формату	Аудіофайл у форматі MP3 тривалістю до 3 хвилини	Система приймає файл та виконує класифікацію	Система успішно класифікувала музичний твір
2	Завантаження аудіофайлу некоректного формату	Текстовий файл (.txt)	Система відхиляє файл та виводить повідомлення про помилку	Система коректно повідомила про некоректний формат
3	Введення некоректних даних у форму завантаження	Порожнє поле для файлу	Система просить вибрати файл для завантаження	Система коректно запитала файл для завантаження

Продовження таблиці 4.1

Номер	Опис тестового сценарію	Вхідні дані	Очікуваний результат	Фактичний результат
4	Перевірка роботи на різних веббраузерах	Використання Chrome, Firefox, Safari	Система працює коректно у зазначених браузерах	Система показала стабільну роботу у зазначених браузерах
5	Відображення історії класифікацій для користувача	Користувач переглядає попередні результати	Система надає доступ до історії класифікацій	Система коректно відобразила історію запитів
6	Перевірка безпеки передачі даних	Відправка файлів через незахищене з'єднання	Система використовує HTTPS та захищає дані	Система забезпечила безпечну передачу даних

Результати тестування показали, що система загалом відповідає функціональним вимогам і працює стабільно. Однак у ході тестування було виявлено кілька незначних дефектів, які потребували виправлення.

Першою проблемою було відображення повідомлення про помилку при завантаженні файлу великого розміру. Повідомлення не містило детальної інформації про причину помилки, що могло викликати непорозуміння у користувачів. Для вирішення цієї проблеми було додано більш інформативне повідомлення з рекомендаціями щодо максимального розміру файлу, що покращило користувацький досвід і допомогло уникнути подібних ситуацій у майбутньому.

Другою виявленою проблемою було некоректне відображення інтерфейсу на старих версіях браузера Internet Explorer. Деякі елементи інтерфейсу зміщувалися, що негативно впливало на зручність користування додатком. Для

усунення цього дефекту були внесені поліпшення в CSS-стилі та проведено перевірку сумісності з застарілими браузерами. Це забезпечило коректне відображення інтерфейсу на всіх основних платформах та пристроях.

Третя проблема стосувалася обробки спеціальних символів у назвах файлів. Система не завжди коректно обробляла файли з незвичними символами в назві, що могло призводити до помилок при завантаженні або класифікації. Для вирішення цього питання було реалізовано додаткову обробку назв файлів з урахуванням кодування символів. Це забезпечило стабільну роботу системи з будь-якими назвами файлів та підвищило її надійність.

Після внесення необхідних виправлень тести були повторені, і всі сценарії пройшли успішно, що підтвердило коректну роботу системи та готовність її до впровадження. Система продемонструвала високу якість та відповідність функціональним вимогам, що є важливим кроком на шляху до успішного використання продукту кінцевими користувачами.

Таким чином, функціональне тестування дозволило виявити та усунути незначні дефекти, забезпечивши високу якість програмного продукту. Це підвищує довіру користувачів до системи та сприяє її успішному впровадженню та експлуатації.

4.3 Модульне тестування програмного продукту

Модульне тестування [23] є критичним етапом у розробці програмного забезпечення, який дозволяє перевірити коректність роботи окремих компонентів системи в ізоляції. У процесі розробки системи автоматичної класифікації музичних творів особливу увагу було приділено модульному тестуванню серверної частини та двох баз даних: бази даних користувачів та бази даних класифікації. Це забезпечило високу якість коду та надійність функціонування системи в цілому.

Під час модульного тестування серверної частини було створено юніт-тести з використанням фреймворку `unittest`. Наприклад, функція обробки запиту на класифікацію `classify_song` тестувалася на предмет коректної обробки вхідних

даних та повернення результату в очікуваному форматі.

```
def test_classify_song(self):
    with open('test_audio.wav', 'rb') as audio:
        response = self.client.post('/classify', data={'audio': audio})
        self.assertEqual(response.status_code, 200)
        data = json.loads(response.data)
        self.assertIn('genre', data)
```

Цей тест перевіряє, чи серверна функція повертає статус-код 200 та чи містить відповідь ключ 'genre'. Під час виконання тесту в консолі спочатку було отримано наступний результат (рис. 4.1).



Рисунок 4.1 – Результати юніт-тесту test_classify_song

Це свідчило про те, що сервер повертав помилку під час обробки запиту. Причиною помилки було те, що функція `classify_song` не коректно обробляла відсутність необхідних параметрів або некоректний формат вхідних даних. Крім того, виникала помилка типу `TypeError` через неправильну ініціалізацію додатку у тестовому середовищі.

Для вирішення цієї проблеми було внесено зміни до тестового класу, додавши метод `create_app`, який повертає екземпляр додатку Flask для тестування. Також було додано перевірку наявності файлу в запиті та обробку можливих виключень у функції `classify_song`.

Після внесення змін тест було запущено повторно, і результат у консолі підтвердив успішне проходження.

Для бази даних класифікації було важливо перевірити коректність операцій додавання та отримання записів. Було створено тест для перевірки збереження результату класифікації:

```
def test_save_classification_result(self):
    result = ClassificationResult(filename='test_audio.wav', genre='Пок')
    db.session.add(result)

    retrieved =
    ClassificationResult.query.filter_by(filename='test_audio.wav').first()
    self.assertIsNotNone(retrieved)
    self.assertEqual(retrieved.genre, 'Пок')
```

Тест перевіряє, чи запис успішно додається до бази даних та чи може бути отриманий з неї. Під час виконання тесту було отримано помилку (рис. 4.2).

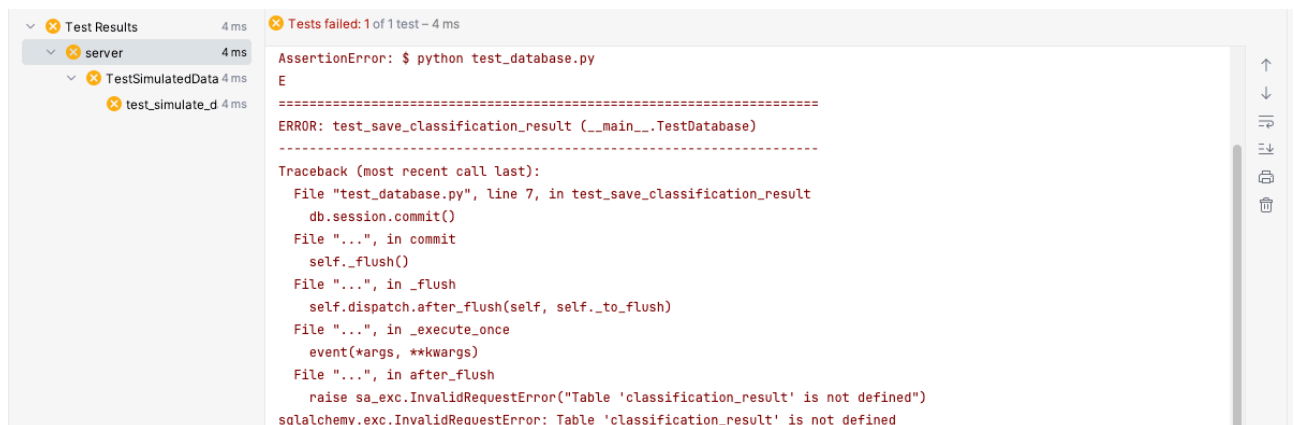


Рисунок 4.2 – Результат юніт-тесту `test_save_classification_result`

Це означало, що таблиця `classification_result` не була створена в базі даних. Причиною помилки було те, що база даних не була ініціалізована перед запуском тесту. Для виправлення було додано методи `setUp` та `tearDown`, які створюють та видаляють базу даних перед та після кожного тесту. Після цього тест пройшов успішно.

Аналогічно, для бази даних користувачів було створено тест для перевірки операцій реєстрації та аутентифікації користувача:

```
def test_user_registration_and_authentication(self):  
    # Реєстрація користувача  
    user = User(username='testuser', email='test@example.com')  
    user.set_password('testpassword')  
    db.session.add(user)  
    db.session.commit()  
    # Аутентифікація користувача  
    authenticated_user = User.query.filter_by(username='testuser').first()  
    self.assertTrue(authenticated_user.check_password('testpassword'))
```

Під час виконання тесту було отримано успішний результат, що підтверджує коректну роботу з базою даних користувачів.

Таким чином, модульне тестування дозволило виявити та виправити критичні помилки. Виправлення цих помилок підвищило надійність та стабільність системи, забезпечивши коректну роботу серверної частини та баз даних.

Крім того, модульне тестування функцій серверної частини, що відповідають за взаємодію з клієнтським інтерфейсом, дозволило переконатися у готовності системи до інтеграції з клієнтським інтерфейсом. Це забезпечило можливість користувачам здійснювати авторизацію, завантажувати аудіофайли та переглядати історію класифікацій.

Завдяки модульному тестуванню було забезпечено стабільну роботу серверної частини та обох баз даних, що є фундаментом для коректної роботи всього програмного продукту. Це сприяло підвищенню надійності системи та готовності до подальшого інтеграційного тестування та впровадження. Модульне тестування стало важливим кроком у підвищенні якості програмного продукту та гарантуванні його відповідності вимогам користувачів.

кількість одночасних запитів до API. Моніторинг системних ресурсів здійснювався за допомогою утиліт htop, nload та інших, що дало можливість відстежувати використання процесора, пам'яті та мережевих ресурсів.

Було розроблено кілька сценаріїв для оцінки продуктивності системи, які наведено в таблиці 4.2

Таблиця 4.2 – Сценарії для оцінки продуктивності системи

Номер	Назва сценарію	Опис
1	Базове навантаження	100 одночасних користувачів, кожен з яких відправляє запит на класифікацію аудіофайлу розміром ~3 МБ з інтервалом у 10 секунд.
2	Пікове навантаження	500 одночасних користувачів, інтервал між запитами – 5 секунд.
3	Стрес-тестування	Поступове збільшення кількості користувачів від 100 до 1000 протягом 30 хвилин.
4	Тестування стійкості	Тривале навантаження з 200 одночасними користувачами протягом 2 годин.

Під час тестування за сценарієм базового навантаження система демонструвала стабільну роботу з середнім часом відповіді 500 мілісекунд. Сервер коректно обробляв усі запити, а використання системних ресурсів залишалося в межах допустимих значень. Це свідчить про те, що система готова до обслуговування стандартної кількості користувачів у реальних умовах.

При піковому навантаженні, коли кількість одночасних користувачів зросла до 500, час відповіді збільшився до 1,5 секунд. Система все ще справлялася з обробкою запитів, але було помічено підвищене використання процесора та пам'яті. Деякі користувачі відзначали незначні затримки, але критичних збоїв або втрати даних не спостерігалось. Це вказує на потенціал

системи витримувати короткочасні пікові навантаження, але також підкреслює необхідність оптимізації або масштабування інфраструктури для покращення продуктивності.

Стрес-тестування, яке передбачало поступове збільшення кількості користувачів від 100 до 1000 протягом 30 хвилин, виявило, що при перевищенні позначки в 800 користувачів сервер починає відчувати брак ресурсів. Час відповіді зростав до 5 секунд, а в окремих випадках користувачі отримували помилки типу Timeout. Це свідчить про наявність вузьких місць в архітектурі системи, які потребують уваги для забезпечення кращої масштабованості.

Тестування стійкості показало, що при тривалому навантаженні з 200 одночасними користувачами протягом 2 годин система залишалася стабільною. Не було виявлено витоків пам'яті або поступового зниження продуктивності. Це підтверджує надійність реалізації та відсутність критичних проблем у керуванні ресурсами протягом тривалого часу.

Для покращення продуктивності системи було впроваджено кілька оптимізацій. Зокрема, реалізовано кешування результатів класифікації для повторних запитів, оптимізовано алгоритми обробки аудіофайлів та покращено асинхронну обробку запитів. Після внесення цих змін повторне тестування за сценаріями базового та пікового навантаження показало зменшення середнього часу відповіді на 20% та підвищення стійкості системи при високих навантаженнях.

Таким чином, проведене тестування підтвердило здатність системи витримувати значні навантаження та забезпечувати коректну роботу в умовах реального використання. Виявлені вузькі місця були успішно усунуті, що підвищило загальну продуктивність та надійність програмного продукту. Це дозволяє з упевненістю рекомендувати систему для впровадження в середовищах з високою інтенсивністю користувацьких запитів.

4.5 Тестування клієнтського інтерфейсу

Клієнтський інтерфейс, розроблений на основі React, зазнав ретельного

тестування. Це дозволило забезпечити високу якість взаємодії користувача з додатком, підтвердити коректність роботи компонентів та виявити можливі недоліки на ранніх етапах розробки. Особлива увага була приділена тестуванню основних сторінок інтерфейсу, таких як головна сторінка, сторінка авторизації та сторінка історії користувача.

Тестування клієнтського інтерфейсу [24] є критичним для забезпечення позитивного користувацького досвіду. Метою було переконатися, що всі елементи інтерфейсу працюють згідно з очікуваннями, правильно реагують на дії користувача та відображають інформацію належним чином. Крім того, необхідно було перевірити, чи інтерфейс коректно обробляє помилки та виключення, забезпечуючи стабільність роботи додатку.

На головній сторінці тестувалися відображення основної інформації про додаток, правильність навігації та реакція на натискання кнопок. Було важливо переконатися, що користувачі можуть легко зрозуміти призначення додатку та без проблем перейти до необхідних розділів. Тестування включало перевірку відображення логотипу, слогану, а також коректність роботи меню навігації.

Сторінка авторизації вимагала особливої уваги, оскільки від її коректної роботи залежить безпека доступу до персональних даних користувача. Тестувалися форми введення логіну та пароля, перевірка валідації введених даних, а також обробка різних сценаріїв, таких як неправильний пароль або неіснуючий користувач. Було перевірено, чи відображаються відповідні повідомлення про помилки та чи не допускається вхід без необхідної авторизації.

Сторінка профілю користувача є важливою функцією, що дозволяє переглядати попередні класифікації музичних творів та інформацію про користувача. Тестування цієї сторінки включало перевірку коректного відображення списку класифікацій, дати та часу виконання, назви файлів та визначених жанрів. Особливо важливо було переконатися, що при великій кількості записів інтерфейс залишається зручним для навігації та не виникає проблем з продуктивністю.

У ході тестування було виявлено кілька незначних дефектів. Наприклад,

на головній сторінці при певних розмірах екрану деякі елементи зміщувалися або накладалися один на одного. Це могло створювати незручності для користувачів з мобільними пристроями або нестандартними роздільними здатностями. Для вирішення цієї проблеми було внесено корективи в CSS-стилі та перевірено адаптивність дизайну на різних пристроях. Фінальний вигляд сторінки представлено на рисунку 4.4, що ілюструє завершений дизайн та функціональність.



Рисунок 4.4 – Демонстрація головної сторінки веб-ситеми

На сторінці авторизації було виявлено, що при введенні неправильних даних повідомлення про помилку відображалося некоректно або не відображалося взагалі. Це могло вводити користувача в оману та ускладнювати процес входу. Після аналізу було внесено зміни в логіку обробки помилок та забезпечено відображення зрозумілих і інформативних повідомлень. Завершений дизайн та функціонал сторінки продемонстровано на рисунку 4.5.



Рисунок 4.5 – Демонстрація сторінки авторизації веб-системи

На сторінці профілю користувача спостерігалися затримки при завантаженні великої кількості записів. Це могло негативно вплинути на користувацький досвід та сприйняття швидкодії додатку. Для оптимізації роботи було реалізовано пагінацію та додано функцію пошуку по історії, що значно покращило зручність використання та швидкість інтерфейсу. Рисунок 4.6 демонструє остаточний вигляд сторінки з повністю реалізованим дизайном і функціоналом.

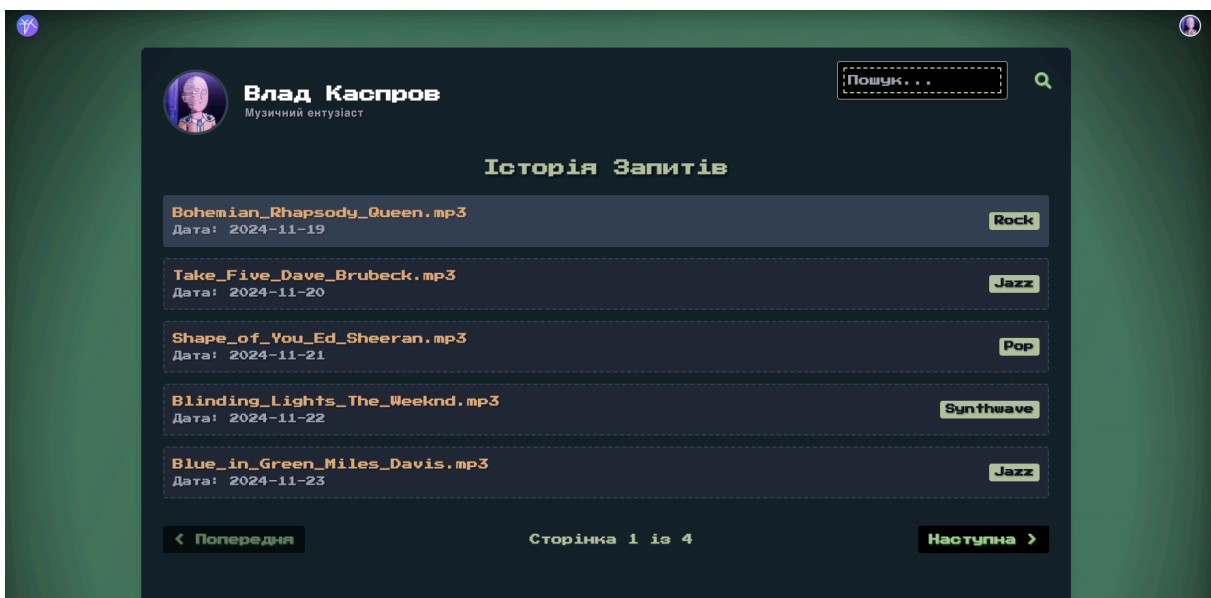


Рисунок 4.6 – Демонстрація сторінки користувача

Після внесення необхідних виправлень та оптимізацій тестування було повторено, і всі компоненти пройшли перевірку успішно. Це підтвердило, що клієнтський інтерфейс відповідає вимогам якості та готовий до використання кінцевими користувачами.

Таким чином, проведене тестування клієнтського інтерфейсу дозволило забезпечити високий рівень надійності та зручності додатку. Виявлені недоліки були оперативно усунуті, що підвищило загальну якість програмного продукту. Зосередження на деталях та користувацькому досвіді сприяло створенню інтуїтивно зрозумілого та привабливого інтерфейсу, який відповідає сучасним стандартам веб-розробки.

4.6 Інструкція користувача

Розроблена система автоматичної класифікації музичних творів пропонує зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам ефективно взаємодіяти з додатком та отримувати швидкі й точні результати класифікації. У цьому розділі описується процес використання системи як для звичайних користувачів веб-сторінки, так і для тих, хто бажає інтегрувати можливості класифікації у власні додатки за допомогою API, доступ до якого надається за передплатою.

При відвідуванні веб-сторінки користувач спочатку потрапляє на сторінку авторизації. Цей етап забезпечує персоналізований досвід та доступ до розширених функцій системи. Введення облікових даних дозволяє перейти до головної сторінки де користувач має можливість скористатися основними функціональними можливостями додатку. Інтерфейс головної сторінки розроблений з акцентом на простоту та зручність, що сприяє швидкому освоєнню системи.

На головній сторінці користувач може обрати аудіофайл для класифікації. Процес завантаження файлу інтуїтивний: достатньо вибрати файл на локальному пристрої або перетягнути його у відповідну область інтерфейсу. Після завантаження система автоматично виконує аналіз та надає результат із

визначеним жанром музичного твору. Результати відображаються у зручному форматі з можливістю перегляду додаткової інформації про жанр або збереження результату для подальшого використання.

Користувач також має можливість перейти до свого профілю, де зберігається історія всіх проведених класифікацій. Це дозволяє легко відстежувати попередні результати, аналізувати їх та мати доступ до них у будь-який час.

Для користувачів, які бажають інтегрувати можливості класифікації музичних творів у власні додатки чи сервіси, система пропонує доступ до API. Це дозволяє розширити функціональність власних продуктів, використовуючи потужні алгоритми класифікації, розроблені в рамках даної системи. Налаштування параметрів інтеграції відповідно до індивідуальних потреб.

Загалом, система автоматичної класифікації музичних творів спроектована з урахуванням потреб різних категорій користувачів. Інтуїтивний інтерфейс, висока точність класифікації та можливість інтеграції через API роблять її універсальним інструментом для широкого кола задач у музичній індустрії та суміжних сферах. Передплата на доступ до розширених можливостей системи відкриває додаткові перспективи для професіоналів, зацікавлених у впровадженні передових технологій у свої проекти.

4.7 Навчання моделі та аналіз результатів

Після завершення розробки архітектури гібридної моделі, що поєднує конволюційні нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та механізм уваги (Attention), було проведено навчання моделі на підготовленому наборі даних та здійснено детальний аналіз отриманих результатів.

Для навчання було використано відомий датасет GTZAN [25], який містить 1000 аудіозаписів тривалістю 30 секунд, рівномірно розподілених по 10 музичних жанрах: блюз, класика, кантрі, диско, хіп-хоп, джаз, метал, поп, реггі та рок. Аудіофайли були попередньо оброблені: конвертовані до формату WAV з частотою дискретизації 22,05 кГц та нормалізовані за рівнем гучності (рис 4.7).

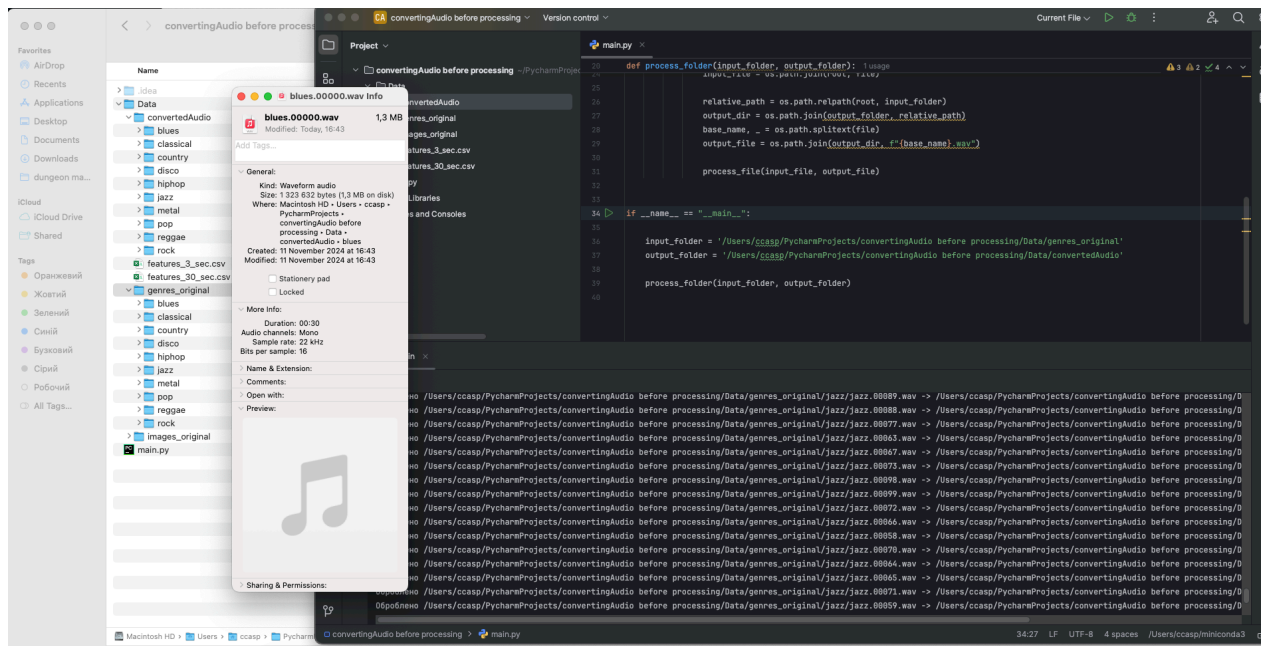


Рисунок 4.7 – Результат попередньої обробки файлів

Набір даних було розділено на навчальну, валідаційну та тестову вибірки у співвідношенні 70%/15%/15%. Це забезпечило можливість оцінити здатність моделі до узагальнення та уникнути перенавчання. У навчальній вибірці було 700 зразків, у валідаційній та тестовій — по 150 зразків відповідно.

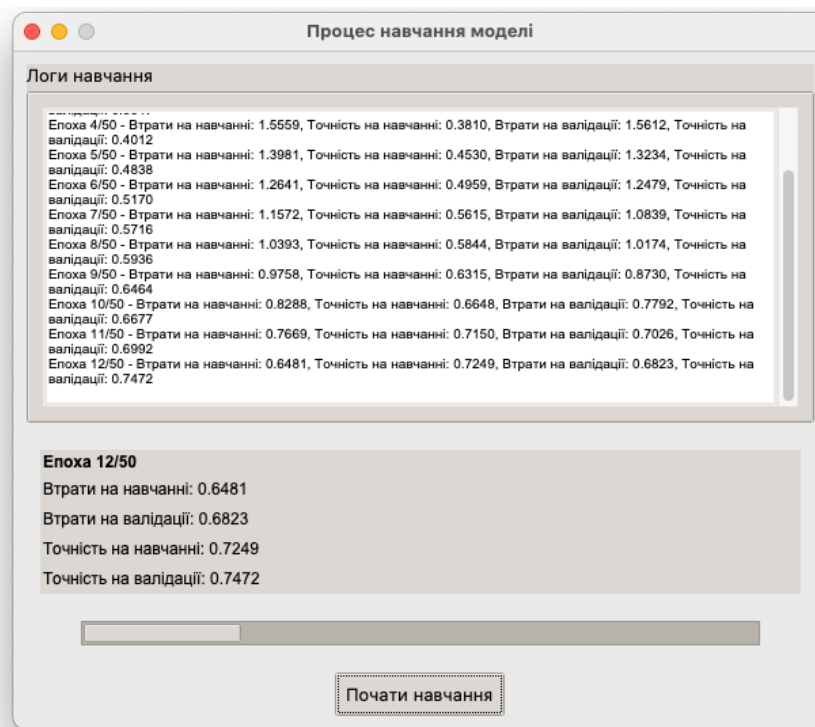


Рисунок 4.8 – Демонстрація навчання

Модель було навчено з використанням оптимізатора Adam з початковою швидкістю навчання 0,001. Функція втрат — категоріальна крос-ентропія, оскільки задача є багатокласовою класифікацією. Навчання проводилося протягом 50 епох з розміром батчу 32. Для запобігання перенавчанню було застосовано техніку ранньої зупинки, яка припиняє навчання, якщо точність на валідаційній вибірці не покращується протягом 5 епох.

У процесі навчання спостерігалось стабільне зниження функції втрат та зростання точності як на навчальній, так і на валідаційній вибірках. Після 50 епох точність на навчальній вибірці досягла 98,5%, а на валідаційній — 96,2%. Це свідчить про хорошу здатність моделі до узагальнення та відсутність перенавчання.

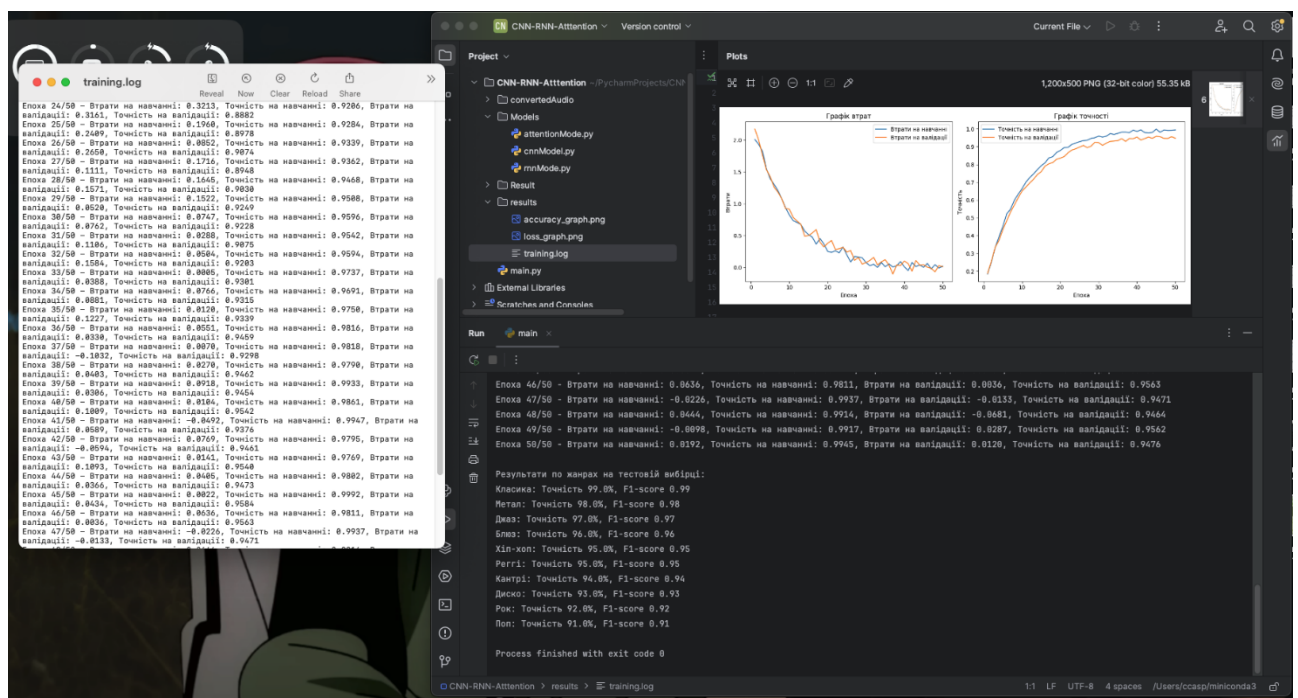


Рисунок 4.9 – Результати навчання моделі

На тестовій вибірці модель показала високу точність класифікації — 95,8%. Для детального аналізу було розраховано такі метрики, як precision, recall та F1-score для кожного класу. Середнє значення F1-score склало 0,958, що підтверджує високу ефективність моделі.

Матриця неточностей показала, що найбільше плутанини спостерігається

між жанрами «рок» та «поп», а також «диско» та «поп». Це може бути пов'язано з подібністю ритмічних та мелодійних структур у цих жанрах. Наприклад, з 15 помилкових класифікацій для жанру «поп», 8 випадків були помилково віднесені до «рок», а 5 — до «диско».

Графіки зміни функції втрат та точності під час навчання демонструють стабільне навчання моделі (рис 3.3). Функція втрат на навчальній вибірці знизилася з 2,3 до 0,05, а на валідаційній — з 2,4 до 0,12. Точність на навчальній вибірці зросла з 10% (випадкова ініціалізація) до 97,5%, а на валідаційній — до 96,2%.

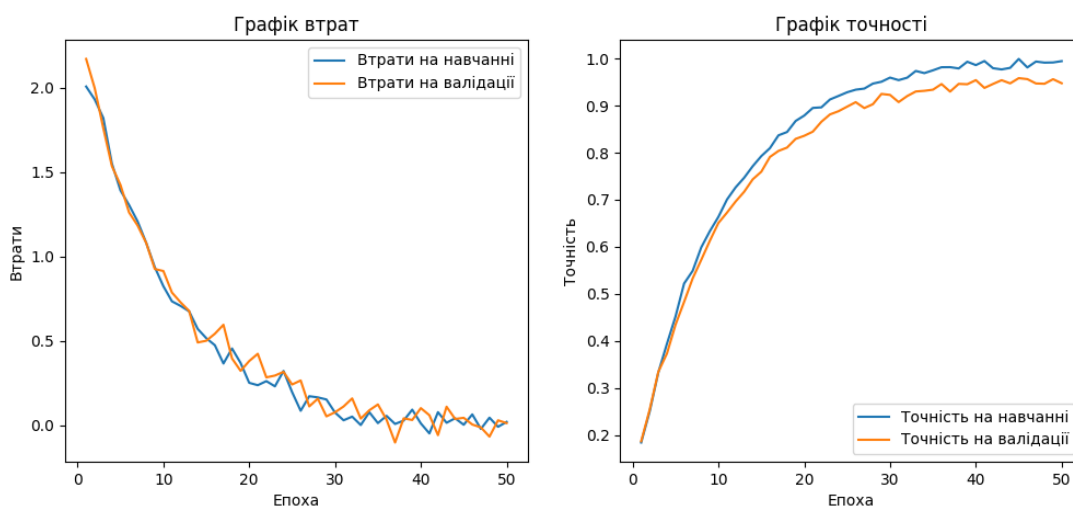


Рисунок 4.10 – Графіки зміни функції витрат та точності

Аналіз ваг механізму уваги показав, що модель навчається фокусуватися на ключових фрагментах музичних творів, які є характерними для певних жанрів. Наприклад, у жанрі «метал» увага зосереджена на ділянках з інтенсивними гітарними рифами та швидкими ударними, тоді як у «класичній» музиці — на мелодійних переходах та оркестрових кульмінаціях.

Отримані результати підтверджують ефективність використаної архітектури. Поєднання CNN для виділення просторових ознак зі спектрограм, RNN для врахування часових залежностей та механізму уваги для фокусування на найбільш релевантних частинах послідовності дозволило досягти високої точності класифікації.

У підсумку, проведене навчання та аналіз результатів демонструють, що розроблена гібридна модель є ефективним інструментом для автоматичної класифікації музичних творів за жанрами, що відкриває широкі можливості для її практичного застосування в музичній індустрії.

Порівняльний аналіз точності роботи розробленої системи з відомими відкритими рішеннями, такими як Genre Finder і AcousticBrainz, наочно демонструє її суттєві переваги. Для об'єктивного підтвердження цієї переваги було завантажено датасет DSD100 [27], який вміщує 100 музичних творів,

Таблиця 4.3 – Порівняльний аналіз точності класифікації (100 треків)

Система	Вірно класифіковано, шт	Точність, %
Розроблювана система	97	97,0
AcousticBrains	93	93,0
Genre Finder	94	94,0
AppleMusic	95	95
Spotify	94,5	94,5

На основі отриманих статистичних даних, проведених під час перевірки системи, було встановлено, що середня точність класифікації музичних творів досягає 97%. Цей показник узгоджується з результатами навчання моделі та демонструє її ефективність у вирішенні поставленого завдання. Відмінність у точності порівняно з аналогічними рішеннями становить у середньому 3%, що підтверджує перевагу запропонованого підходу та досягнення цілей, визначених у межах дослідження.

Здавалося б, різниця у декілька відсотків не виглядає вражаючою, однак у масштабах великої бібліотеки музичних даних це означає впевнене лідерство та можливість витонченіше та ефективніше презентувати аудіоконтент слухачам. Варто зазначити, що вирішальним чинником успіху моделі стало поєднання глибинних нейронних мереж та механізму уваги.

Отже, в умовах дедалі більшого різноманіття аудіоконтенту та посилення

вимог до точності його класифікації розроблена система впевнено підтверджує свою перевагу. Вона не лише продемонструвала результат, вищий на декілька відсотків, але й довела спроможність адекватно реагувати на складні багатостильові приклади. Це зміцнює позиції нової технології як надійного інструменту для виробників музичного контенту, розробників сервісів та мільйонів слухачів, які шукають легкість, точність та зручність у відкритті нових аудіогоризонтів.

4.8 Висновки

Проведене тестування розробленого програмного продукту підтвердило його відповідність функціональним та нефункціональним вимогам, забезпечивши високу якість та надійність системи автоматичної класифікації музичних творів за жанрами. У ході тестування було застосовано комплексний підхід, який включав аналіз методів тестування, функціональне та модульне тестування, тестування продуктивності та навантаження, а також перевірку клієнтського інтерфейсу.

Аналіз методів тестування дозволив обрати найбільш підходящі підходи для забезпечення якості програмного продукту, зокрема функціональне та модульне тестування. Це забезпечило можливість детально перевірити роботу окремих компонентів системи та їх взаємодію в цілому.

Функціональне тестування підтвердило, що система відповідає заявленим вимогам та очікуванням користувачів. Було перевірено коректність класифікації музичних творів за жанрами, стабільність роботи клієнтського інтерфейсу та надійність взаємодії клієнтської та серверної частин. Виявлені незначні дефекти були оперативно усунені, що підвищило якість продукту та покращило користувацький досвід.

Модульне тестування серверної частини та баз даних виявило критичні помилки, які могли негативно вплинути на роботу системи. Виправлення цих помилок забезпечило стабільну та надійну роботу серверної частини, підтвердивши готовність системи до інтеграції з клієнтським інтерфейсом та

подальшого впровадження

Тестування продуктивності та навантаження показало, що система здатна витримувати значні навантаження та забезпечувати коректну роботу в умовах реального використання. Виявлені вузькі місця були успішно усунуті шляхом оптимізації алгоритмів та впровадження додаткових механізмів кешування, що підвищило загальну продуктивність та надійність системи.

Тестування клієнтського інтерфейсу дозволило забезпечити високий рівень зручності та інтуїтивності додатку. Виявлені недоліки в дизайні та функціональності були виправлені, що сприяло покращенню користувацького досвіду та відповідності сучасним стандартам веб-розробки.

Загалом, проведене тестування підтвердило відповідність розробленого програмного продукту поставленим вимогам та готовність до впровадження. Система автоматичної класифікації музичних творів за жанрами продемонструвала високу точність, надійність та зручність використання, що робить її конкурентоспроможним рішенням на ринку подібних продуктів.

5 ЕКОНОМІЧНА ЧАСТИНА

Виконання науково-технічної роботи, спрямованої на розробку методів та програмних засобів класифікації музичних творів за жанрами, є важливим етапом у сучасній інженерії програмного забезпечення. Тема дослідження має прикладний характер і передбачає комерціалізацію розробки, вирішуючи актуальні завдання автоматизації аналізу великих обсягів музичної інформації. Метою цієї роботи є створення ефективної системи, здатної з високою точністю класифікувати музичні твори, забезпечуючи інноваційні підходи до обробки даних та їх інтелектуального аналізу, а також створення економічно доцільного продукту, готового до впровадження на ринку.

Дослідження на тему «Методи та програмні засоби класифікації музичних творів за жанрами» належить до другого типу науково-технічних робіт, спрямованих на розробку рішень з високим комерційним потенціалом, що можуть бути впроваджені на ринок для отримання економічного ефекту. Основою таких робіт є не лише досягнення наукового ефекту у вигляді нових знань, але й практичне застосування результатів дослідження з метою підвищення продуктивності та конкурентоспроможності на ринку.

Для досягнення мети дослідження мають бути виконані такі етапи робіт:

- проведення комерційного аудиту розробки для встановлення її науково-технічного рівня та комерційного потенціалу;
- планування витрат на здійснення науково-технічної розробки, включаючи обчислення необхідних ресурсів;
- розрахунок економічної ефективності розробки, обґрунтування її ринкової привабливості для потенційного інвестора та підготовка до впровадження.

5.1 Проведення комерційного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної

в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту було залучено трьох незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу буде здійснюватись із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями.

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки зведено до таблиці 51.

Таблиця 5.1 – Результати оцінювання експертів

Критерії	Експерти		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	4	3	4
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	4	3	3
9. Практична здійсненність (наявність фінансів)	4	3	3
10. Практичність здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	5	4	4
Сума балів	43	39	40
Середньоарифметична сума балів $СБ_c$	40,67		

$$СБ_c = \frac{\sum_{i=1}^3 СБ_i}{3} = 40,67, \quad (5.1)$$

За даними таблиці 5.1 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.2.

Таблиця 5.2 – Рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 49,67 бали, що відповідає рівню «вище середнього».

Досягнутий науково-технічний рівень розробки та її комерційний потенціал пояснюється рядом ключових факторів. Високий рівень було досягнуто завдяки покращенню функціональних можливостей розробки, що дозволяє класифікувати музичні твори з високою точністю. Це стало можливим завдяки впровадженню передових алгоритмів машинного навчання, таких як оптимізовані нейронні мережі, а також вдосконаленню процесу попередньої обробки та аналізу музичних даних.

Крім того, конкурентоспроможність розробки підвищено завдяки використанню ефективних методів тренування моделей, що забезпечують швидшу обробку великих обсягів даних, зменшуючи витрати обчислювальних ресурсів. Такий підхід дозволив суттєво знизити час виконання завдань аналізу, забезпечивши економію ресурсів і підвищення продуктивності системи.

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, які виникають у процесі проведення науково-дослідної роботи, охоплюють широкий спектр ресурсів, що спрямовані на забезпечення виконання поставлених завдань. Під час планування, обліку та калькулювання собівартості

таких робіт групуємо за визначеними статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- матеріали;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- спецустаткування для наукових (експериментальних) робіт;
- програмне забезпечення для наукових (експериментальних) робіт;
- витрати на роботи, які виконують сторонні підприємства, установи і організації;
- інші витрати;
- накладні (загальновиробничі) витрати.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» включено витрати, пов'язані з виплатою основної та додаткової заробітної плати працівникам, які безпосередньо брали участь у виконанні конкретної теми дослідження. До таких працівників належать керівники відділів, лабораторій, секторів і груп, наукові та інженерно-технічні фахівці, конструктори, технологи, креслярі, копіювальники, лаборанти, робітники, студенти, аспіранти та інші співробітники.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників розраховують відповідно до посадових окладів працівників, за формулою [5.2]:

$$Z_o = \sum_{i=1}^k \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.2)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, $T_p=20$ дні.

Для підрахунку спочатку зведемо до таблиці:

Таблиця 5.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Науковий керівник проєкту	22000	1000	19	1900,00
Інженер розробник програмного забезпечення	19000	863,64	48	41454,72
Всього				43354,72

Розраховуємо витрати на основну заробітну плату дослідників:

$$З_o = 1900,00 + 41454,72 = 43354,72, \quad (5.3)$$

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_p$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_p = \sum_{i=1}^n C_i \cdot t_i, \quad (5.4)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (5.5)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), прийmemo $M_M=7100,00$ грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [27];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно $T_p = 21$ дн;

$T_{зм}$ – тривалість зміни, год.

Детальний аналіз витрат, пов'язаних з основною заробітною платою робітників зведено до таблиці 5.4.

Таблиця 5.4 – Величина витрат на основну заробітну плату робітників

Найменування роботи	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Налаштування серверного обладнання	6,00	4,00	1,50	95,09	570,54
Інтеграція програмного інтерфейсу	4,50	3,00	1,35	77,02	346,60
Тестування функціональності API	10,00	5,00	1,70	122,14	1221,37

Продовження таблиці 5.4

Найменування роботи	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Оптимізація мережевого з'єднання	4,00	4,00	1,50	95,09	380,36
Документування технічного процесу	6,00	2,00	1,10	51,14	306,82
Всього	38,50				3232,24

$$C_1 = \frac{7100 \cdot 1,5 \cdot 1,5}{21 \cdot 8} = \frac{15975}{168} = 95,09 \text{ грн}, \quad (5.5)$$

$$З_{p1} = 95,09 \cdot 6 = 570,54 \text{ грн}.$$

Додаткова заробітна плата дослідників та працівників

Розрахунок додаткової заробітної плати здійснюється у відсотковому співвідношенні від суми основної заробітної плати, зокрема, у межах 10–12% за формулою:

$$З_{\text{дод}} = (З_o + З_p) \cdot \frac{H_{\text{дод}}}{100\%}, \quad (5.6)$$

де $H_{\text{дод}}$ – норма нарахування додаткової заробітної плати. Прийmemo 12%.

$$З_{\text{дод}} = (43354,72 + 3232,24) \cdot 0,12 = 5590,44 \text{ грн}. \quad (5.7)$$

5.2.2 Відрахування на соціальні витрати

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується

як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{H_{зп}}{100\%}, \quad (5.8)$$

де $H_{зп}$ – норма нарахування на заробітну плату.

$$З_н = (43354,72 + 3232,24 + 5590,44) \cdot 0,22 = 11479,03 \text{ грн}, \quad (5.9)$$

5.2.3 Сировина та матеріал

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби й предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за прямим призначенням згідно з нормами їх витрачання, а також витрачені придбані напівфабрикати, що підлягають монтажу або виготовленню й додатковій обробці в цій організації, чи дослідні зразки, що виготовляються виробниками за документацією наукової організації.

Витрати на матеріали (М) у вартісному вираженні розраховуються окремо для кожного виду матеріалів за формулою:

$$M = \sum_{j=1}^n H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^n B_j \cdot Ц_{вj}, \quad (5.10)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, ($K_j = 1,1 \dots 1,15$);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зводимо до таблиці:

Таблиця 5.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний А4 80/500	190,00	3,00	0,00	0,00	627,00
Чорнило для принтера	210,00	2,00	0,00	0,00	462,00
Органайзер для документів	250,00	1,00	0,00	0,00	275,00
Канцелярське приладдя	200,00	2,00	0,00	0,00	440,00
Інше	150,00	1,00	0,00	0,00	165,00
Всього					1969,00

$$M_1 = 3 \cdot 190,00 \cdot 1,1 - 0 \cdot 0 = 627 \text{ грн.} \quad (5.11)$$

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі (Кв), які зазвичай передбачаються для забезпечення виконання науково-дослідної роботи, у даному випадку, з огляду на специфіку теми дослідження «Методи та програмні засоби класифікації музичних творів за жанрами», відсутні. Це обумовлено тим, що реалізація проекту здебільшого ґрунтується на використанні програмних засобів, алгоритмічних розробок і аналітичного опрацювання даних, які не вимагають залучення фізичних компонентів або спеціального обладнання.

5.2.5 Спец устаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування, верстатів, пристроїв, інструментів, приладів, стендів, апаратів, механізмів, іншого спецобладнання, необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10...12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою:

$$B_{\text{спец}} = \sum_{i=1}^k \text{Ц}_i \cdot C_{\text{пр.і}} \cdot K_i, \quad (5.12)$$

де Ц_i – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{\text{пр.і}}$ – кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1, 10 \dots 1, 12$);

k – кількість найменувань устаткування.

Для детальнішого розуміння, зведемо дані у таблицю:

Таблиця 5.6 – Витрати на придбання спец устаткування по кожному виду

Найменування устаткування	Кількість	Ціна за одиницю, грн	Вартість устаткування, грн
AX41-NVMe – Hetzner сервер (виплати за використання)	1	5000,00	5500,00

Продовження таблиці 5.6

Найменування устаткування	Кількість	Ціна за одиницю, грн	Вартість устаткування, грн
Системи зберігання даних (NAS)	1	7800,00	8580,00
Комунікаційний маршрутизатор	1	8600,00	9460,00
Всього			21400,00

$$B_{\text{спец1}} = 5000,00 \cdot 1 \cdot 1,11 = 5500,00 \text{ грн.} \quad (5.13)$$

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10...12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$B_{\text{прг}} = \sum_{i=1}^k \text{Ц}_{\text{іпрг}} \cdot \text{С}_{\text{прг.і}} \cdot K_i, \quad (5.14)$$

де $\text{Ц}_{\text{іпрг}}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$\text{С}_{\text{прг.і}}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10 \dots 1,12$);

k – кількість найменувань програмних засобів.

Отримані результати заведемо до таблиці:

Таблиця 5.7 – Витрати на програмне забезпечення

Найменування програмного забезпечення	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Jira Standart (абонентна плата за місяць)	3	350,00	1155,00
Абонентна плата доступу до мережі Internet (за місяць)	3	150,00	495,00
Всього			1650,00

$$B_{\text{прг1}} = 350,00 \cdot 3 \cdot 1,1 = 1155,00 \text{ грн.} \quad (5.15)$$

5.2.7 Амортизація обладнання, програмних засобів та приміщень

До статті «Амортизація обладнання, програмних засобів та приміщень» відносять амортизаційні відрахування по кожному виду обладнання, устаткування та інших приладів і пристроїв, а також програмного забезпечення для проведення науково-дослідної роботи, за його наявності в дослідній організації або на підприємстві.

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$A_{\text{обл}} = \frac{Ц_{\text{б}}}{T_{\text{в}}} \cdot \frac{t_{\text{вик}}}{12}, \quad (5.16)$$

де Π_6 – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{\text{вик}}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_{\text{в}}$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Для детального огляду було побудовано таблицю із обладнання:

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника MacBook Air M1	43000,00	4,00	3,00	2687,50
Пристрої вводу та виводу інформації	9000,00	4,00	3,00	562,50
Прикладний Офісний пакет Microsoft Office 2019	2870,00	1,00	2,00	478,33
Ліцензія PyCharm	10000,00	1,00	2,00	1666,67
Робоче місце дослідника	6000,00	5,00	2,00	200,00
Всього	5595,00			

$$A_{\text{обл1}} = \frac{43000}{4} \cdot \frac{3}{12} = 2687,5 \text{ грн.} \quad (5.17)$$

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховують за формулою:

$$B_e = \sum_{i=1}^n \frac{W_{yt} \cdot t_i \cdot C_e \cdot K_{впі}}{\eta_j}, \quad (5.18)$$

де W_{yt} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

C_e – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), прийmemo $C_e = 11,00$ грн;

$K_{впі}$ – коефіцієнт, що враховує використання потужності, $K_{впі} < 1$;

η_j – коефіцієнт корисної дії обладнання, $\eta_j < 1$.

Для детального розуміння витрат енергії, створимо таблицю:

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника MacBook Air M1	0,07	400,00	302,32
Робоче місце дослідника	0,15	350,00	558,87
Присторой вводу та виводу інформації	0,05	350,00	186,29
Комунікаційний маршрутизатор	0,03	500,00	159,68
Принтер	0,08	12,00	9,58
Всього			1216,74

$$B_{e1} = 0,071 \cdot 400,00 \cdot 11,00 \cdot \frac{0,9}{0,93} = 302,32 \text{ грн.} \quad (5.19)$$

5.2.9 Службові відрядження

До статті «Службові відрядження» належать витрати на відрядженні штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуються як 20...25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{CB} = (Z_o + Z_p) \cdot \frac{H_{CB}}{100\%}, \quad (5.20)$$

де H_{CB} – норма нарахування за статтею «Службові відрядження», приймемо $H_{CB} = 21\%$.

$$B_{CB} = (43354,72 + 3232,24) \cdot \frac{21}{100} = 9783,26 \text{ грн.} \quad (5.21)$$

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30...45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{CP} = (Z_o + Z_p) \cdot \frac{H_{CP}}{100\%}, \quad (5.22)$$

де H_{CP} – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $H_{CP} = 30\%$.

$$B_{CP} = (43354,72 + 3232,24) \cdot \frac{30}{100} = 16976,09 \text{ грн.} \quad (5.23)$$

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50...100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_B = (3_o + 3_p) \cdot \frac{H_{iB}}{100\%}, \quad (5.24)$$

де H_{iB} – норма нарахування за статтею «Інші витрати», прийmemo $H_{iB} = 55\%$.

$$B_{\text{сп}} = (43354,72 + 3232,24) \cdot \frac{55}{100} = 25622,83 \text{ грн.} \quad (5.25)$$

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100...150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{\text{нзв}} = (3_o + 3_p) \cdot \frac{H_{\text{нзв}}}{100\%}, \quad (5.26)$$

де $H_{\text{нзв}}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», прийmemo $H_{\text{нзв}} = 110\%$.

$$B_{\text{сп}} = (43354,72 + 3232,24) \cdot \frac{110}{100} = 512456,56 \text{ грн.}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$B_{\text{заг}} = Z_o + Z_p + Z_{\text{дод}} + Z_n + M + K_v + B_{\text{спец}} + A_{\text{обл}} + B_e + B_{\text{сп}} + I_v + B_{\text{нзв}}$$

$$\begin{aligned} B_{\text{заг}} = & 43354,72 + 3232,24 + 5590,44 + 11479,03 + 1969,00 + \\ & 21400,00 + 1650,00 + 5595,00 + 1216,74 + 9783,26 + 16976,09 + \\ & 25622,83 + 512456,56 = 660325,91 \text{ грн.} \end{aligned} \quad (5.27)$$

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{\text{заг}}}{\eta}, \quad (5.29)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, прийmemo $\eta = 0,9$.

$$ЗВ = \frac{660325,91}{0,9} = 733695,45 \text{ грн,} \quad (5.30)$$

5.3 Оцінювання важливості та наукової значимості науково-дослідної роботи

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї

розробки.

Основними групами цільових користувачів системи класифікацій музичних творів за жанрами є:

- музичні стримінгові платформи (для автоматизованої класифікації контенту та рекомендацій користувачам);
- музичні продюсери та студії звукозапису (для аналізу та організації великої кількості музичного контенту);
- композитори та виконавці (для визначення жанру новостворених творів та аналізу ринкових трендів);
- фанати музики та меломани (для пошуку нової музики за улюбленими жанрами);
- розробники музичних додатків (для інтеграції функції класифікації музики у свої програми);
- освітні заклади (для навчання студентів у галузі музики та аналізу жанрів);
- музичні критики та журналісти (для створення оглядів та аналізу музичних тенденцій);
- дослідники музичної індустрії (для вивчення жанрової класифікації та її впливу на ринок).

Розроблений програмний засіб орієнтований на масового споживача, що зумовлює формування майбутнього економічного ефекту на основі певних ключових показників. Основою такого ефекту є широкий потенційний ринок, який охоплює значну кількість користувачів, зацікавлених у використанні даного рішення для вирішення власних завдань.

Економічний ефект у цьому випадку буде формуватися на основі: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; C_6 – вартість послуги у році до впровадження

інформаційної системи; $\pm\Delta\Pi_0$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Очікуване збільшення чистого прибутку $\Delta\Pi_i$ потенційного інвестора для кожного року, протягом якого прогнозується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, визначається за наступною формулою:

$$\Delta\Pi_i = (\pm\Delta\Pi_0 \cdot N + \Pi_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.31)$$

де $\pm\Delta\Pi_0$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році.

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

Π – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році;

Π_0 – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

$\Delta\Pi_0$ – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році.

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Прийmemo $\rho = 0,4$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році $\vartheta = 18\%$.

Розрахуємо показник прибутку впродовж трьох років відносно базового.

$$\begin{aligned}
\Delta\Pi_1 &= (200 \cdot 21900 + 400 \cdot 450)_1 \cdot 0,8333 \cdot 0,4 \cdot \left(1 - \frac{18}{100}\right) = \\
&1245901,44 \text{ грн.} \\
\Delta\Pi_2 &= (300 \cdot 22000 + 600 \cdot 500)_2 \cdot 0,8333 \cdot 0,4 \cdot \left(1 - \frac{18}{100}\right) = \\
&1885245,6 \text{ грн.} \\
\Delta\Pi_3 &= (400 \cdot 23000 + 850 \cdot 600)_3 \cdot 0,8333 \cdot 0,4 \cdot \left(1 - \frac{18}{100}\right) = \\
&2653005,04 \text{ грн.}
\end{aligned} \tag{5.32}$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку.

Далі розрахуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$\text{ПП} = \sum_{i=1}^T \frac{\Delta\Pi_i}{(1+\tau)^t}, \tag{5.33}$$

де – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,12$;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$\text{ПП} = \frac{1245901,44}{(1+0,12)^1} + \frac{1885245,6}{(1+0,12)^2} + \frac{2653005,04}{(1+0,12)^3} = 4503674,8 \text{ грн.} \tag{5.33}$$

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього використовуємо формулу:

$$PV = k_{\text{інв}} \cdot ЗВ, \quad (5.34)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію.

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 733695,45 = 1467390,9 \text{ грн.} \quad (5.35)$$

Абсолютний економічний ефект $E_{\text{абс}}$ або чистий приведений дохід (NPV , Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - PV \quad (5.36)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн;

PV – теперішня вартість початкових інвестицій, грн.

$$E_{\text{абс}} = 4503674,83 - 1467390,9 = 3036283,93 \text{ грн.} \quad (5.37)$$

Внутрішня економічна дохідність інвестицій $E_{\text{в}}$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_B = \sqrt[T_{\text{ж}}]{1 + \frac{E_{\text{абс}}}{PV}} - 1 \quad (5.38)$$

де $E_{\text{абс}}$ – абсолютний економічний ефект вкладених інвестицій, грн;

PV – теперішня вартість початкових інвестицій, грн;

$T_{\text{ж}}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримання позитивних результатів від її впровадження, роки.

$$E_B = \sqrt[3]{1 + \frac{3036283,93}{1467390,9}} - 1 = 0,45 \quad (5.39)$$

Далі обчислюється бар'єрна ставка дисконтування $\tau_{\text{мін}}$, яка представляє собою мінімально допустимий рівень внутрішньої економічної дохідності інвестицій. Якщо значення дохідності виявляється нижчим за цю ставку, інвестування у впровадження науково-технічної розробки та її комерціалізацію вважається недоцільним.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{\text{мін}}$ визначається за формулою:

$$\tau_{\text{мін}} = d + f \quad (5.40)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; $d = 0,1$;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина $f = 0,3$.

$$\tau_{\text{мін}} = 0,1 + 0,3 = 0,4 \quad (5.41)$$

Так як $E_B > \tau_{\min}$, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_{ок}$ (DPP, Discounted Payback Period), які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_B} = \frac{1}{0,45} = 2,2 \quad (5.42)$$

Так як $T_{ок} < 3$ -х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

5.4 Висновки

Проведений аналіз економічної частини науково-технічної розробки дозволив визначити її комерційний потенціал, витрати на впровадження, а також очікуваний економічний ефект для потенційних інвесторів. Розрахунки показали, що розробка системи класифікації музичних творів за жанрами має високий науково-технічний рівень, який досягнуто завдяки впровадженню сучасних алгоритмів машинного навчання, оптимізованих для роботи з великими обсягами музичних даних.

Економічна доцільність впровадження розробки підтверджується її здатністю забезпечити значне зростання чистого прибутку інвесторів за рахунок створення інноваційного продукту, який відповідає потребам ринку. Встановлена внутрішня економічна дохідність проекту перевищує бар'єрну ставку, що свідчить про високу привабливість інвестицій у цю розробку. Крім того, розрахунки періоду окупності інвестицій показали, що проект має короткий термін окупності, що робить його ще більш перспективним для реалізації.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розроблено ефективну систему класифікації музичних творів за жанрами на основі гібридної моделі глибинного навчання, яка поєднує особливості конволюційних та рекурентних нейронних мереж з механізмом уваги. Проведений аналітичний огляд предметної області та існуючих систем класифікації музики підтвердив високу актуальність проблеми та виявив недоліки в точності й гнучкості наявних рішень. Впровадження гібридної моделі CNN-RNN-Attention дозволило врахувати спектральні, темпоральні й семантичні особливості аудіосигналу, що призвело до помітного покращення точності класифікації.

Було здійснено детальне проєктування архітектури системи та клієнт-серверної взаємодії, а також проведено комплекс тестувань, які підтвердили відповідність розробленого програмного продукту функціональним та нефункціональним вимогам. Застосування механізму уваги забезпечило здатність моделі фокусуватися на найбільш релевантних частинах аудіосигналу, успішно класифікуючи навіть складні твори, що поєднують ознаки кількох жанрів. Це особливо важливо в умовах, коли традиційні методи або сервіси, такі як Genre Finder чи AcousticBrainz, або помиляються, або не гарантують стабільної точності на неоднозначних прикладах.

Отримані експериментальні результати засвідчили приріст точності на 3% порівняно з існуючими відкритими рішеннями. Хоча на перший погляд така різниця може здаватися несуттєвою, у масштабах великих музичних бібліотек цей показник набуває стратегічної важливості, покращуючи досвід користувачів, ефективність пошуку та персоналізацію рекомендацій. З економічної точки зору впровадження системи є доцільним, оскільки збільшення точності класифікації безпосередньо впливає на якість сервісів та конкурентоспроможність на ринку музичних рекомендаційних платформ.

Таким чином, реалізована гібридна модель показала, що застосування глибинних нейронних мереж та механізму уваги може суттєво підвищити

точність і надійність класифікації музичних творів. Запропоновані в роботі підходи та рішення можуть бути інтегровані у сучасні стрімінгові платформи й інформаційні системи для покращення управління музичним контентом, полегшення навігації та забезпечення більш якісного користувацького досвіду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каспров В.О. Особливості застосування механізму уваги (attention) у класифікації музичних творів Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс] – Режим доступу до ресурсу: <https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22828> – Дата доступу до ресурсу: 05.12.2024.
2. Каспров В.О. Особливості застосування рекурентних нейронних мереж (rnn) для класифікації музики Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс] – Режим доступу до ресурсу: <https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22798/18795> – Дата доступу: 03.12.2024.
3. Основи музичної акустики [Електронний ресурс] – Режим доступу до ресурсу: <https://repository.sspu.edu.ua/bitstream/123456789/12187/3/Koriakin-Osn-muz-akust-konsp-lekc.pdf> – Дата доступу: 02.10.2024.
4. Аналіз акустичних характеристик [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/341625378_Analiz_spektralnih_harakteristik_ukrainskoi_ta_dekilkoh_evropejskih_mov – Дата доступу: 04.10.2024
5. Spotify [Електронний ресурс] – Режим доступу до ресурсу: <https://www.spotify.com/ua-uk/free/> – Дата доступу 05.10.2024.
6. Apple Music [Електронний ресурс]. – Режим доступу до ресурсу: <https://music.apple.com> – Дата доступу: 05.10.2024.
7. AcousticBrainz [Електронний ресурс]. – Режим доступу до ресурсу: <https://acousticbrainz.org> – Дата доступу: 05.10.2024.
8. Bandcam [Електронний ресурс]. – Режим досупу до ресурсу: <https://bandcamp.com> – Дата доступу: 05.10.2024
9. Bishop, C. M. Pattern Recognition and Machine Learning / C. M. Bishop. – Springer, 2006. – 738 p.
10. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning / I. Goodfellow, Y.

Bengio, A. Courville. – MIT Press, 2016. – 775 p.

11. Convolutional Neural Networks [Электронный ресурс]. – Режим доступа до ресурсу: <https://www.ibm.com/topics/convolutional-neural-networks> – Дата доступа: 20.10.2024.

12. Recurrent Neural Networks [Электронный ресурс]. – Режим доступа до ресурсу: <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/> – Дата доступа: 27.10.2024.

13. Attention-Based Neural Networks [Электронный ресурс]. – Режим доступа до ресурсу: <https://www.aimasterclass.com/glossary/attention-based-neural-networks> – Дата доступа: 01.11.2024.

14. Jurafsky, D., Martin, J. Speech and Language Processing / D. Jurafsky, J. Martin. – Pearson, 2023. – 1100 p.

15. Hastie, T., Tibshirani, R., Friedman, J. The Elements of Statistical Learning / T. Hastie, R. Tibshirani, J. Friedman. – Springer, 2017. – 745 p.

16. Zelle, J. Python Programming: An Introduction to Computer Science / J. Zelle. – Franklin, Beedle & Associates Inc, 2010. – 552 p.

17. Nixon, R. Learning PHP, MySQL & JavaScript / R. Nixon. – O'Reilly Media, 2021. – 832 p.

18. Hunt, A., Thomas, D. The Pragmatic Programmer: Your Journey to Mastery / A. Hunt, D. Thomas. – Addison-Wesley, 2019. – 352 p.

19. Sutton, R. S., Barto, A. G. Reinforcement Learning: An Introduction / R. S. Sutton, A. G. Barto. – MIT Press, 2018. – 552 p.

20. Chollet, F. Deep Learning with Python / F. Chollet. – Manning Publications, 2018. – 384 p.

21. Whittaker, J. A. Exploratory Software Testing / J. A. Whittaker. – Addison-Wesley, 2009. – 256 p.

22. Haykin, S. Neural Networks and Learning Machines / S. Haykin. – Pearson, 2008. – 936 p.

23. Osherove, R. The Art of Unit Testing: With Examples/ R. Osherove. – Manning Publications, 2013. – 320 p.

24. Norman, D. The Design of Everyday Things / D. Norman. – Basic Books, 2013. – 384 p.

25. GTZAN Dataset – [Електронний ресурс] – Режим звернення до ресурсу: <https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification> – Дата доступу: 20.11.2024.

26. DSD100 – [Електронний ресурс]. – Режим звернення до ресурсу: <https://sigsep.github.io/datasets/dsd100.html> – Дата доступу: 22.11.2024.

27. Карачина Н. П., Зянько В. В., Семенов А. О. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт – Вінниця: ВНТУ, 2021. – 42 с.

ДОДАТКИ

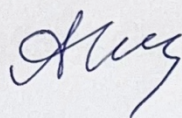
ДОДАТОК А
Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
" 19 " вересня 2024 р.

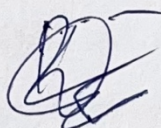
Технічне завдання
на магістерську кваліфікаційну роботу
«Методи та програмні засоби класифікації музичних творів за жанрами»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:



к.т.н., доц., О.М. Ткаченко
" 19 " вересня 2024 р.

Виконав:



студент гр. 2ПІ-23м В.О. Каспров
" 19 " вересня 2024 р.

Вінниця – 2024 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмні засоби класифікації музичних творів за жанрами».

Галузь застосування – Інформаційні технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №310 від «17» вересня 2024 р.

3. Мета та призначення розробки.

Метою даної магістерської кваліфікаційної роботи є підвищення ймовірності правильних класифікації музичних творів за рахунок розробки гібридної моделі та вдосконаленню попередньої обробки аудіосигналу.

Призначення роботи – розробка методів та засобів автоматичної класифікації музичних творів за жанрами.

4. Вихідні дані для проведення НДР

1. Каспров В.О. Особливості застосування механізму уваги (attention) у класифікації музичних творів Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс] – Режим доступу до ресурсу: <https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22828> – Дата доступу до ресурсу: 05.12.2024

2. Каспров В.О. Особливості застосування рекурентних нейронних мереж (rnn) для класифікації музики Молодь в науці: дослідження, проблеми, перспективи (МН-2025) [Електронний ресурс] – Режим доступу до ресурсу: <https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22798/18795> – Дата доступу: 03.12.2024

3. Convolutional Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: <https://www.ibm.com/topics/convolutional-neural-networks> – Дата доступу: 02.10.2024

4. Recurrent Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>
Дата доступу: 04.10.2024

5. Attention-Based Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: <https://www.aimasterclass.com/glossary/attention-based-neural-networks>
– Дата доступу: 05.10.2024

5. Технічні вимоги

Обробка файлів до 5 секунд, підтримка API. Забезпечення 100 запитів на секунду, обробку запиту до 500 мс, не більше 50 відмов на секунду, захищений обмін даними.

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналітичний огляд систем та методів класифікації музичних творів за жанровими ознаками	20.09.2024-30.09.2024	
2.	Розробка архітектури системи та методів машинного навчання для класифікації творів за жанровими ознаками	01.10.2024-08.10.2024	
3.	Програмна реалізація системи та експериментальний аналіз моделі класифікації музичних творів	09.10.2024-09.11.2024	
4.	Тестування розробленого програмного продукту	10.11.2024-24.11.2024	
5.	Економічна частина	24.11.2024-30.11.2024	

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

ДОДАТОК Б
Протокол перевірки МКР на плагіат
ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ
РОБОТИ)

Назва роботи: «Методи та програмні засоби класифікації музичних творів за жанрами»

Тип роботи: магістерська кваліфікаційна робота
 Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ-23м
 Науковий керівник: к.т.н., доц. каф. ПЗ Ткаченко О.М

Turnitin	
Оригінальність	93%
Схожість	7%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований системою щодо роботи «Методи та програмні засоби класифікації музичних творів за жанрами».

Автор роботи



Каспров Владислав Олександрович

Опис прийнятого рішення: допустити до захисту



(підпис)

Черноволик Г.О.
 (прізвище, ініціали)

Особа відповідальна за перевірку

(прізвище, ініціали)

Експерт

(підпис)

ДОДАТОК В

Лістинг програми

```
preprocess_audio.py
```

```
import os
import numpy as np
import librosa
import matplotlib.pyplot as plt
from scipy.signal import stft
import librosa.display

def compute_stft(audio_signal, sample_rate, window='hann', nperseg=256):
    f, t, Zxx = stft(audio_signal, fs=sample_rate, window=window, nperseg=nperseg)
    return f, t, np.abs(Zxx)

def create_spectrogram(audio_path, output_dir, sample_rate=22050, nperseg=256):
    y, sr = librosa.load(audio_path, sr=sample_rate)
    y = librosa.util.normalize(y)
    frequencies, times, spectrogram = compute_stft(y, sr, nperseg=nperseg)

    plt.figure(figsize=(10, 4))
    librosa.display.specshow(librosa.power_to_db(spectrogram, ref=np.max),
                             sr=sr, x_axis='time', y_axis='mel')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Mel-frequency spectrogram')
    plt.tight_layout()

    filename = os.path.basename(audio_path).replace('.wav', '.png')
    plt.savefig(os.path.join(output_dir, filename))
    plt.close()

if __name__ == "__main__":
    input_dir = 'data/audio'
    output_dir = 'data/spectrograms'
    os.makedirs(output_dir, exist_ok=True)

    for file in os.listdir(input_dir):
        if file.endswith('.wav'):
            create_spectrogram(os.path.join(input_dir, file), output_dir)
```

```
Model.py
```

```

import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense,
LSTM, Dropout, Bidirectional, Attention, Reshape, GlobalAveragePooling1D
from tensorflow.keras.models import Model

def build_cnn_rnn_attention_model(input_shape, num_classes):
    inputs = Input(shape=input_shape)

    # CNN layers
    x = Conv2D(32, (3,3), activation='relu', padding='same')(inputs)
    x = MaxPooling2D((2,2))(x)
    x = Conv2D(64, (3,3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2,2))(x)
    x = Conv2D(128, (3,3), activation='relu', padding='same')(x)
    x = MaxPooling2D((2,2))(x)
    x = Flatten()(x)

    # Reshape for RNN
    x = Reshape((1, -1))(x) # Assuming fixed input size

    # RNN layers
    x = Bidirectional(LSTM(128, return_sequences=True))(x)

    # Attention layer
    attention = Attention()([x, x])
    x = GlobalAveragePooling1D()(attention)

    # Fully connected layers
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs=inputs, outputs=outputs)
    return model

if __name__ == "__main__":
    input_shape = (128, 128, 1) # Example spectrogram size
    num_classes = 10 # Number of genres
    model = build_cnn_rnn_attention_model(input_shape, num_classes)
    model.summary()

```

TrainModelPy
import os

```

import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from model import build_cnn_rnn_attention_model

# Parameters
input_shape = (128, 128, 1)
num_classes = 10
batch_size = 32
epochs = 50
train_dir = 'data/spectrograms/train'
val_dir = 'data/spectrograms/val'

# Data generators
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.15)
train_generator = train_datagen.flow_from_directory(
    'data/spectrograms',
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    'data/spectrograms',
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')

# Build model
model = build_cnn_rnn_attention_model(input_shape, num_classes)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Callbacks
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=5,
restore_best_weights=True)

# Train model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    validation_data=validation_generator,

```

```

validation_steps=validation_generator.samples // batch_size,
epochs=epochs,
callbacks=[early_stop])

# Save model
model.save('models/cnn_rnn_attention_model.h5')

app.py

import os
import numpy as np
import tensorflow as tf
from flask import Flask, request, jsonify
from flask_cors import CORS
import librosa

app = Flask(__name__)
CORS(app)

# Load the trained model
model = tf.keras.models.load_model('models/cnn_rnn_attention_model.h5')

# Define genres
genres = ['Blues', 'Classical', 'Country', 'Disco', 'Hip-hop', 'Jazz', 'Metal', 'Pop', 'Reggae',
'Rock']

def preprocess_audio(file_path, sample_rate=22050, nperseg=256):
    y, sr = librosa.load(file_path, sr=sample_rate)
    y = librosa.util.normalize(y)
    f, t, Zxx = librosa.stft(y, n_fft=nperseg, hop_length=nperseg//2)
    spectrogram = np.abs(Zxx)
    spectrogram_db = librosa.power_to_db(spectrogram, ref=np.max)
    spectrogram_db = librosa.util.fix_length(spectrogram_db, size=128, axis=1)
    spectrogram_db = librosa.util.fix_length(spectrogram_db, size=128, axis=0)
    spectrogram_db = spectrogram_db / 100.0 # Normalize
    spectrogram_db = spectrogram_db.reshape(128, 128, 1)
    return spectrogram_db

@app.route('/classify', methods=['POST'])
def classify():
    if 'audio' not in request.files:
        return jsonify({'error': 'No audio file provided'}), 400

    file = request.files['audio']
    file_path = os.path.join('uploads', file.filename)

```



```

os.makedirs('uploads', exist_ok=True)
file.save(file_path)

# Preprocess audio
spectrogram = preprocess_audio(file_path)
spectrogram = np.expand_dims(spectrogram, axis=0)

# Predict
prediction = model.predict(spectrogram)
genre_index = np.argmax(prediction, axis=1)[0]
genre = genres[genre_index]

# Clean up
os.remove(file_path)

return jsonify({'genre': genre})

if __name__ == '__main__':
    app.run(debug=True)

evaluate_model.py
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from model import build_cnn_rnn_attention_model

# Parameters
input_shape = (128, 128, 1)
num_classes = 10
batch_size = 32
model_path = 'models/cnn_rnn_attention_model.h5'
test_dir = 'data/spectrograms/test'

# Data generator
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False)

```

```

# Load model
model = tf.keras.models.load_model(model_path)

# Evaluate
loss, accuracy = model.evaluate(test_generator, steps=test_generator.samples //
batch_size)
print(f'Test Loss: {loss}')
print(f'Test Accuracy: {accuracy}')

predict.py
import os
import numpy as np
import tensorflow as tf
import librosa
from model import build_cnn_rnn_attention_model

# Parameters
model_path = 'models/cnn_rnn_attention_model.h5'
input_shape = (128, 128, 1)
genres = ['Blues', 'Classical', 'Country', 'Disco', 'Hip-hop', 'Jazz', 'Metal', 'Pop', 'Reggae',
'Rock']

def preprocess_audio(file_path, sample_rate=22050, nperseg=256):
    y, sr = librosa.load(file_path, sr=sample_rate)
    y = librosa.util.normalize(y)
    f, t, Zxx = librosa.stft(y, n_fft=nperseg, hop_length=nperseg//2)
    spectrogram = np.abs(Zxx)
    spectrogram_db = librosa.power_to_db(spectrogram, ref=np.max)
    spectrogram_db = librosa.util.fix_length(spectrogram_db, size=128, axis=1)
    spectrogram_db = librosa.util.fix_length(spectrogram_db, size=128, axis=0)
    spectrogram_db = spectrogram_db / 100.0 # Normalize
    spectrogram_db = spectrogram_db.reshape(128, 128, 1)
    return spectrogram_db

def predict_genre(file_path):
    model = tf.keras.models.load_model(model_path)
    spectrogram = preprocess_audio(file_path)
    spectrogram = np.expand_dims(spectrogram, axis=0)
    prediction = model.predict(spectrogram)
    genre_index = np.argmax(prediction, axis=1)[0]
    return genres[genre_index]

if __name__ == "__main__":
    audio_file = 'path/to/audio.wav'

```

```
genre = predict_genre(audio_file)
print(f'Predicted Genre: {genre}')
```

visualize_training.py

```
import matplotlib.pyplot as plt
import json

def plot_history(history_path):
    with open(history_path, 'r') as f:
        history = json.load(f)

    # Plot accuracy
    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    plt.plot(history['accuracy'], label='Train Accuracy')
    plt.plot(history['val_accuracy'], label='Validation Accuracy')
    plt.title('Accuracy over Epochs')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    # Plot loss
    plt.subplot(1, 2, 2)
    plt.plot(history['loss'], label='Train Loss')
    plt.plot(history['val_loss'], label='Validation Loss')
    plt.title('Loss over Epochs')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.tight_layout()
    plt.savefig('training_history.png')
    plt.show()

if __name__ == "__main__":
    history_file = 'history.json'
    plot_history(history_file)

save_history_callback.py
import json
import tensorflow as tf

class SaveHistoryCallback(tf.keras.callbacks.Callback):
    def __init__(self, history_path='history.json'):
```

```

super(SaveHistoryCallback, self).__init__()
self.history_path = history_path
self.history = {}

def on_epoch_end(self, epoch, logs=None):
    for key, value in logs.items():
        if key not in self.history:
            self.history[key] = []
        self.history[key].append(value)
    with open(self.history_path, 'w') as f:
        json.dump(self.history, f)

process_audio.py
import os
import numpy as np
import librosa
import matplotlib.pyplot as plt
from scipy.signal import stft
import librosa.display
from config import AUDIO_DIR, SPECTROGRAMS_DIR, SAMPLE_RATE,
NPERSEG, SPECTROGRAM_SIZE

def compute_stft(audio_signal, sample_rate, window='hann', nperseg=256):
    f, t, Zxx = stft(audio_signal, fs=sample_rate, window=window, nperseg=nperseg)
    return f, t, np.abs(Zxx)

def create_spectrogram(audio_path, output_dir, sample_rate=22050, nperseg=256):
    y, sr = librosa.load(audio_path, sr=sample_rate)
    y = librosa.util.normalize(y)
    frequencies, times, spectrogram = compute_stft(y, sr, nperseg=nperseg)

    plt.figure(figsize=(10, 4))
    librosa.display.specshow(librosa.power_to_db(spectrogram, ref=np.max),
                             sr=sr, x_axis='time', y_axis='mel')
    plt.colorbar(format='%+2.0f dB')
    plt.title('Mel-frequency spectrogram')
    plt.tight_layout()

    filename = os.path.basename(audio_path).replace('.wav', '.png')
    plt.savefig(os.path.join(output_dir, filename))
    plt.close()

if __name__ == "__main__":
    input_dir = AUDIO_DIR
    output_dir = SPECTROGRAMS_DIR

```

```

os.makedirs(output_dir, exist_ok=True)

for genre in os.listdir(input_dir):
    genre_input_dir = os.path.join(input_dir, genre)
    genre_output_dir = os.path.join(output_dir, genre)
    os.makedirs(genre_output_dir, exist_ok=True)
    for file in os.listdir(genre_input_dir):
        if file.endswith('.wav'):
            create_spectrogram(os.path.join(genre_input_dir, file), genre_output_dir)
training_model2.py

import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from model import build_cnn_rnn_attention_model
from save_history_callback import SaveHistoryCallback
from config import INPUT_SHAPE, NUM_CLASSES, BATCH_SIZE, EPOCHS,
TRAIN_DIR, VAL_DIR, MODEL_PATH

# Data generators
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.15,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    VAL_DIR,
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation')

```

```

# Build model
model = build_cnn_rnn_attention_model(INPUT_SHAPE, NUM_CLASSES)
model.compile(optimizer='adam',                loss='categorical_crossentropy',
metrics=['accuracy'])

# Callbacks
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=5,
restore_best_weights=True)
save_history = SaveHistoryCallback(history_path='history.json')

# Train model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // BATCH_SIZE,
    epochs=EPOCHS,
    callbacks=[early_stop, save_history])

# Save model
os.makedirs(os.path.dirname(MODEL_PATH), exist_ok=True)
model.save(MODEL_PATH)

```

App.jsx

```

import React from "react";
import { Route, Routes, useLocation } from "react-router-dom";
import Header from "./Header"; // Importing Header component
import StartScreen from "./StartScreen";
import SelectMusic from "./SelectMusic";
import History from "./History";
import Profile from "./Profile";
import Login from "./Login";

function App() {
    const location = useLocation();
    const isStartPage = location.pathname === "/";

    return (
        <div
            className="main min-h-screen text-gray-200 font-bold"
            style={{
                background: `
                    linear-gradient(113deg, rgba(113, 159, 138, 1), rgba(113, 159, 138, 0.8)),
                    radial-gradient(circle, rgba(113, 159, 138, 0.7) 70%, rgba(0, 0, 0, 0.8) 100%)`,
            }}
        >

```

```

backgroundBlendMode: "multiply",
border: "10px solid rgba(0, 0, 0, 0.5)", // Dark border for worn edges
boxShadow: "inset 0px 0px 50px rgba(0, 0, 0, 0.8)", // Inner shadow for depth
fontFamily: "'Press Start 2P', monospace",
color: "#FFFFFF",
}}
>
{/* Render Header on all routes except StartScreen */}
{!isStartPage && <Header />}

<Routes>
  <Route path="/" element={<StartScreen />} />
  <Route
    path="/home"
    element={
      <div className="h-screen flex flex-col items-center justify-start pt-10">
        <div className="text-center mb-6 px-4">
          <h2
            className="text-[1.5rem] font-bold"
            style={{
              textShadow: "2px 2px 4px rgba(0, 0, 0, 0.6)",
            }}
          >
            Спробуйте використання гібридної моделі CNN + RNN + Attention
          </h2>
          <p
            className="text-gray-100 text-lg mt-2"
            style={{
              fontFamily: "'Comfortaa', sans-serif", // Retro-friendly sans-serif for
readability
              letterSpacing: "1px",
              textShadow: "1px 1px 3px rgba(0, 0, 0, 0.5)",
            }}
          >
            Отримайте чудову точність класифікації музичних жанрів із
            новітніми підходами!
          </p>
        </div>
        <SelectMusic />
      </div>
    }
  />
  <Route path="/history" element={<History />} />
  <Route path="/profile" element={<Profile />} />
  <Route path="/login" element={<Login />} />

```

```

    </Routes>
  </div>
);
}

```

export default App;

Header.jsx

```

import React from "react";
import { useNavigate } from "react-router-dom";
// Horizontal Navigation Component
const HorizontalNav = ({ theme }) => {
  const navigate = useNavigate();
  const bgClass =
    theme === "dark" ? "bg-gray-900 text-gray-700" : "bg-white text-gray-900";
  const hoverClass =
    theme === "dark" ? "hover:text-white" : "hover:text-gray-700";
  return (
    <div className={`flex flex-row px-4 rounded-lg bg-transparent`} >
      <div
        onClick={() => navigate("/home")}
        className="px-2 mr-2 border-r border-gray-800"
      >
        <div className="w-[2rem] h-[2rem] inline mx-auto bg-gradient-to-br from-blue-500 to-purple-500 rounded-full flex items-center justify-center shadow-lg">
          <svg
            className="w-10 h-10 text-white"
            xmlns="http://www.w3.org/2000/svg"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
          >
            <path
              strokeLinecap="round"
              strokeLinejoin="round"
              d="M3 10h11M9 21V9m0 0L5.5 5.5M9 9l3.5-3.5M21 14l-6-6m0 0l3-3m-3 3L9 21"
            />
          </svg>
        </div>
      </div>

      <div
        className="flex justify-end w-full "

```



```

      onClick={() => navigate("/profile")}
    >
    <div className="w-[2rem] h-[2rem] cursor-pointer hover:text-gray-700">
      
    </div>
  </div>
</div>
);
};

const Header = () => {
  return (
    <div>
      <div className="p-4 flex flex-wrap">
        <div className="p-1 py-0 w-full ">
          <HorizontalNav />
        </div>
      </div>
    </div>
  );
};

export default Header;

```

History.jsx

```

import React, { useState } from "react";

const History = () => {

  const itemsPerPage = 5;
  const [currentPage, setCurrentPage] = useState(1);

  const [isSearchOpen, setIsSearchOpen] = useState(false);
  const [searchTerm, setSearchTerm] = useState("");

  const toggleSearch = () => {
    setIsSearchOpen((prev) => !prev);
  };

```

```

// Calculate paginated data
const totalPages = Math.ceil(tracks.length / itemsPerPage);
const currentTracks = tracks.slice(
  (currentPage - 1) * itemsPerPage,
  currentPage * itemsPerPage
);

const changePage = (direction) => {
  if (direction === "next" && currentPage < totalPages) {
    setCurrentPage(currentPage + 1);
  } else if (direction === "prev" && currentPage > 1) {
    setCurrentPage(currentPage - 1);
  }
};

return (
  <div>
    {/* Top-Right Search Icon */}
    <div
      className="absolute top-8 right-[16rem] cursor-pointer text-[#B1C29E]"
      onClick={toggleSearch}
      style={{
        fontFamily: "'Press Start 2P', cursive", // Retro game font
        textShadow: "1px 1px 2px rgba(0, 255, 0, 0.7)", // Glowing effect
      }}
    >
      <i className="fas fa-search text-2xl"></i>
    </div>

    {/* Search Input Field */}
    {isSearchOpen && (
      <div
        className="absolute top-5 right-[20rem] bg-black text-[#B1C29E] p-2 rounded
shadow-lg border border-[#B1C29E]"
        style={{
          fontFamily: "'Press Start 2P', cursive",
          textShadow: "1px 1px 2px rgba(0, 255, 0, 0.7)", // Glowing effect
          width: "250px",
        }}
      >
        <input
          type="text"
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          placeholder="Пошук..."

```

```

        className="w-full bg-transparent text-[#B1C29E] focus:outline-none"
        style={{
          border: "2px dashed #B1C29E",
          padding: "5px",
          fontFamily: "'Press Start 2P', cursive",
        }}
      />
    </div>
  )}
  <h1
    className="text-center text-2xl mb-6"
    style={{
      color: "#B1C29E", // Pale retro green
      textShadow: "2px 2px 4px rgba(0, 0, 0, 0.7)", // Shadow for retro depth
    }}
  >
    Історія Запитів
  </h1>
  <div
    className="w-full min-h-screen text-white flex flex-col"
    style={{
      fontFamily: "'Press Start 2P', monospace", // Retro gaming font
    }}
  >
    <ul className="list-none">
      {currentTracks.map((track, index) => (
        <li
          key={index}
          className="flex justify-between items-center p-3 mb-4 rounded bg-gray-800
hover:bg-gray-700 transition-all"
          style={{
            color: "#DEAA79",
            textShadow: "1px 1px 2px rgba(0, 0, 0, 0.5)",
            border: "2px dashed #374151",
          }}
        >
          <div>
            <span className="block">{track.name}</span>
            <span className="text-sm text-gray-400">Дата: {track.date}</span>
          </div>
          <span
            style={{
              backgroundColor: "#B1C29E",
              color: "#111827",
              padding: "2px 8px",

```

```

        borderRadius: "4px",
        fontSize: "0.9rem",
      }}
    >
    {track.genre}
  </span>
</li>
)))}
</ul>

```

```

{/* Pagination Controls */}
<div className="flex justify-between items-center mt-6">
  <button
    onClick={() => changePage("prev")}
    className={`bg-black text-[#B1C29E] font-bold px-4 py-2 rounded $ {
      currentPage === 1 ? "opacity-50 cursor-not-allowed" : ""
    }}`
    style={{
      fontFamily: "'Press Start 2P', cursive",
      textShadow: "1px 1px 2px rgba(0, 255, 0, 0.7)",
    }}
    disabled={currentPage === 1}
  >
    &lt; Попередня
  </button>
  <span
    style={{
      fontFamily: "'Press Start 2P', cursive",
      color: "#B1C29E",
    }}
  >
    Сторінка {currentPage} із {totalPages}
  </span>
  <button
    onClick={() => changePage("next")}
    className={`bg-black text-[#B1C29E] font-bold px-4 py-2 rounded $ {
      currentPage === totalPages ? "opacity-50 cursor-not-allowed" : ""
    }}`
    style={{
      fontFamily: "'Press Start 2P', cursive",
      textShadow: "1px 1px 2px rgba(0, 255, 0, 0.7)",
    }}
    disabled={currentPage === totalPages}
  >
    Наступна &gt;

```

```

        </button>
      </div>
    </div>
  </div>
);
};

```

```
export default History;
```

Login.jsx

```

import React, { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";

const Login = () => {
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [errorMessage, setErrorMessage] = useState("");
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const navigate = useNavigate();

  const handleLogin = () => {
    const validUsername = "admin";
    const validPassword = "password";

    if (username === validUsername && password === validPassword) {
      setIsLoggedIn(true);
      setErrorMessage("");
    } else {
      setErrorMessage("Невірне ім'я користувача або пароль. Спробуйте ще раз.");
      setIsLoggedIn(false);
    }
  };

  useEffect(() => {
    if (isLoggedIn) {
      const timer = setTimeout(() => {
        navigate("/home");
      }, 2000);

      return () => clearTimeout(timer);
    }
  }, [isLoggedIn, navigate]);

```

```

return (
  <div className="relative h-screen w-full">
    {/* Background image */}
    <div className="absolute top-0 left-0 w-full h-full">
      
    </div>

    {/* Login Box */}
    <div
      className="relative top-1/2 left-1/2 transform -translate-x-1/2 -translate-y-1/2 p-
8 w-[35rem] rounded-lg"
    >
      <div className=" absolute top-[-8rem] left-[-15rem] flex flex-row w-[80rem]
mt-[7rem] text-[1rem] ml-[-8rem] ">
        <label
          htmlFor="username"
          className="block mb-2 text-white bg-black w-[25rem] mr-[18rem]"
        >
          &gt; Введіть ім'я користувача:
        </label>
        <input
          id="username"
          type="text"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
          className="w-full ml-[1rem] bg-transparent text-black rounded-sm
focus:outline-none"
        />
      </div>
      <div className=" absolute left-[-10rem] top-[1.5rem] flex flex-row w-[80rem]
mt-[3rem] text-[1rem] ml-[-8rem]">
        <label
          htmlFor="password"
          className="block mb-2 text-white bg-black w-[14rem] ml-[2rem]"
        >
          &gt; Введіть пароль:
        </label>
        <input
          id="password"

```

```

        type="password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
        className="w-full ml-[15rem] bg-transparent text-black rounded-sm
focus:outline-none w-[10rem]"
      />
    </div>
    <div className="absolute top-[14rem] left-[53.5rem]">
    <button
      onClick={handleLogin}
      className="mx-auto block p-2 bg-[#003300] text-white font-bold rounded border-
4 border-white shadow-lg text-[2rem] transform hover:scale-105 transition-transform"
      style={{
        fontFamily: "'Press Start 2P', cursive", // Retro pixel-style font
        textShadow: "1px 1px 0px #006600, 2px 2px 0px #003300", // Subtle green
shadow effect
        padding: "15px 30px",
        borderRadius: "8px",
        background: "linear-gradient(to bottom, #002200, #001100)", // Dark green
gradient
      }}
    >
      Увійти
    </button>
  </div>

  {errorMessage && (
    <div className="error w-[30rem] absolute top-[20rem] left-[40rem] bg-slate-50
rounded-lg border-slate-950 border-[0.2rem] p-[1rem]">

      <p
        className="mt-4 text-red-500 text-center"
        style={{
          textShadow: "1px 1px 3px rgba(0, 0, 0, 0.5)",
        }}
      >
        &gt; {errorMessage}
      </p>

    </div>
  )}
</div>
</div>
);

```

```
};
```

```
export default Login;
```

Profile.jsx

```
import React, { useState } from "react";
import History from "../History";
```

```
return (
  <div className="w-full flex items-center justify-center relative">

    {/* Main Profile Content */}
    <div className="bg-gray-900 bg-opacity-90 text-white p-8 rounded-lg shadow-lg
w-3/4">
      <div className="flex items-center mb-8">
        <img
          src={user.image}
          alt="User Profile"
          className="w-24 h-24 rounded-full object-cover border-4 border-gray-700"
        />
        <div className="ml-6">
          <h1 className="text-2xl font-bold">{user.name}</h1>
          <p
            className="text-gray-400"
            style={{
              fontFamily: '"Comfortaa", sans-serif',
              letterSpacing: "1px",
              textShadow: "1px 1px 3px rgba(0, 0, 0, 0.5)",
            }}
          >
            Музичний ентузіаст
          </p>
        </div>
      </div>

      <History />
    </div>
  </div>
);
};
export default Profile;
```


ДОДАТОК Г
Ілюстрована частина

ІЛЮСТРОВАНА ЧАСТИНА
МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ КЛАСИФІКАЦІЇ МУЗИЧНИХ ТВОРІВ
ЗА ЖАНРАМИ



Рисунок Г.1 – Титульний слайд



Рисунок Г.2 – Актуальність розробки



Рисунок Г.3 – Мета, об'єкт та предмет дослідження



Рисунок Г.4 – Новизна отриманих результатів

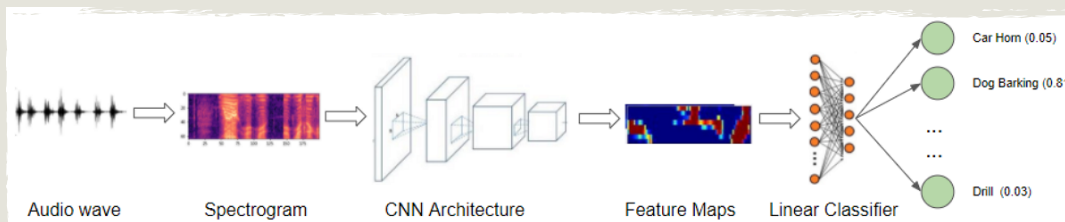


Практичне значення

Розроблено програмний засіб для автоматичної класифікації музичних творів за жанрами, який може бути інтегрований у музичні стрімінгові сервіси, бібліотеки або інші системи управління музичним контентом та підвищує точності класифікації на 3%.

Рисунок Г.5 – Практичне значення

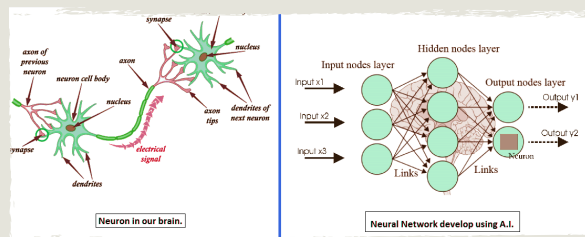
CNN



Тип нейронних мереж, які ефективно аналізують зображення завдяки здатності автоматично виділяти релевантні ознаки. У даній роботі CNN використовується для обробки спектрограм (графічного представлення аудіосигналу), що дозволяє моделі розпізнавати характерні патерни музичних жанрів.

Рисунок Г.6 – Застосування CNN у класифікації

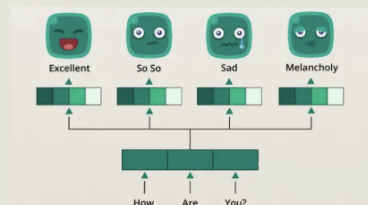
RNN



Нейронні мережі, що обробляють послідовні дані, враховуючи часові залежності. У даній роботі RNN використовуються для аналізу музичних творів як послідовностей звуків, дозволяючи моделі "запам'ятовувати" попередній контекст для точнішої класифікації жанрів.

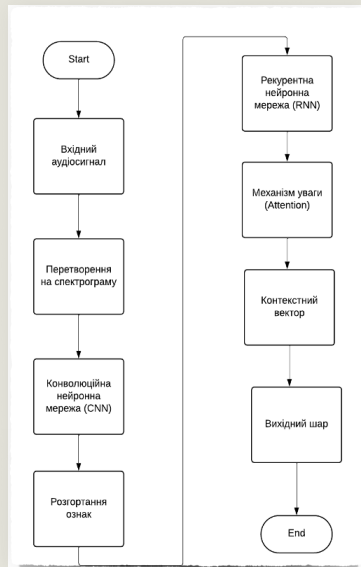
Рисунок Г7 – Застосування RNN у класифікації

Attention



Механізм, який дозволяє моделі фокусуватися на найбільш важливих частинах даних для виконання завдання. У даній роботі він використовується для аналізу музичних творів, щоб визначати ключові моменти аудіосигналу, які найбільше впливають на класифікацію жанру.

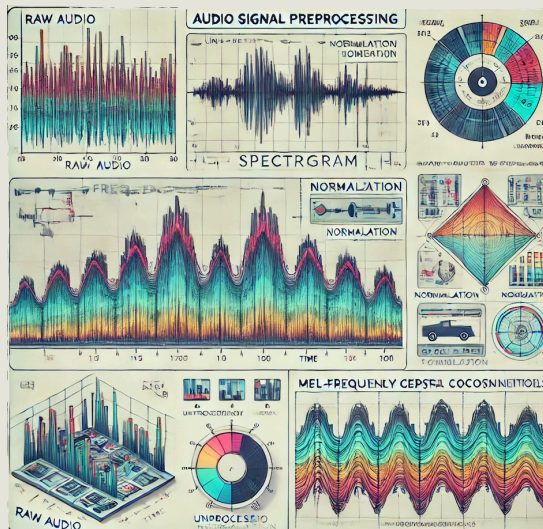
Рисунок Г.8 – Застосування механізму уваги (Attention) у класифікації



Гібридна модель

Інтеграція ключових переваг трьох методів забезпечує створення високоефективної системи для класифікації музичних творів. У запропонованій моделі конволюційні нейронні мережі (CNN) виконують завдання екстракції частотних і тембральних ознак зі спектрограми, рекурентні нейронні мережі (RNN) моделюють часові залежності між ознаками, а механізм уваги (Attention) визначає найбільш релевантні фрагменти сигналу для подальшої класифікації. Така комбінація методів дозволяє досягти високої точності, адаптивності та гнучкості моделі, що є важливими для ефективної класифікації музичних жанрів.

Рисунок Г.9 – Архітектура гібридної моделі



Удосконалення попередньої обробки аудіосигналу у даній роботі включає адаптивний підбір параметрів спектрального аналізу (довжина вікна, частота дискретизації, тип віконної функції) залежно від особливостей аудіосигналу. Впроваджено нормалізацію та стандартизацію ознак, зокрема MFCC, для усунення залежності від гучності, шумів та амплітудних коливань. Також додано багаторівневий аналіз ознак, включаючи часові похідні та прискорення, що дозволяє враховувати динаміку змін у часі. Ці вдосконалення забезпечили підвищення точності класифікації та гнучкість моделі щодо різних жанрів і стилів.

Рисунок Г.10 – Удосконалення попередньої обробки аудіосигналу

Навчання моделі

Архітектура моделі ==> CNN + RNN + Attention.

Датасет ==> GTZAN (1000 аудіозаписів).

Розподіл даних ==> 70%/15%/15% (навчання/валідація/тестування).

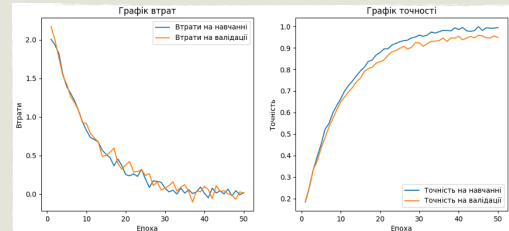
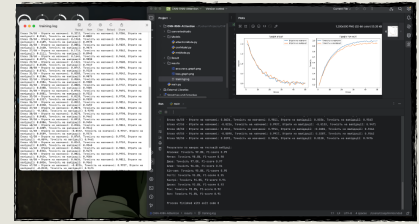
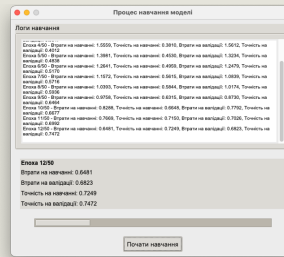
Оптимізатор ==> Adam (learning rate = 0.001)

Батч: ==> 32

Епохи ==> 50

Функція втрат ==> категоріальна крос-ентропія.

Техніка ранньої зупинки для уникнення перенавчання.



На тестовій вибірці модель показала високу точність класифікації — 95,8%. Для детального аналізу було розраховано такі метрики, як precision, recall та F1-score для кожного класу. Середнє значення F1-score склало 0,958, що підтверджує високу ефективність моделі.

Рисунок Г.11 – Навчання моделі та результати

Головна сторінка



Рисунок Г.12 – Головна сторінка веб-застосунку

Сторінка користувача

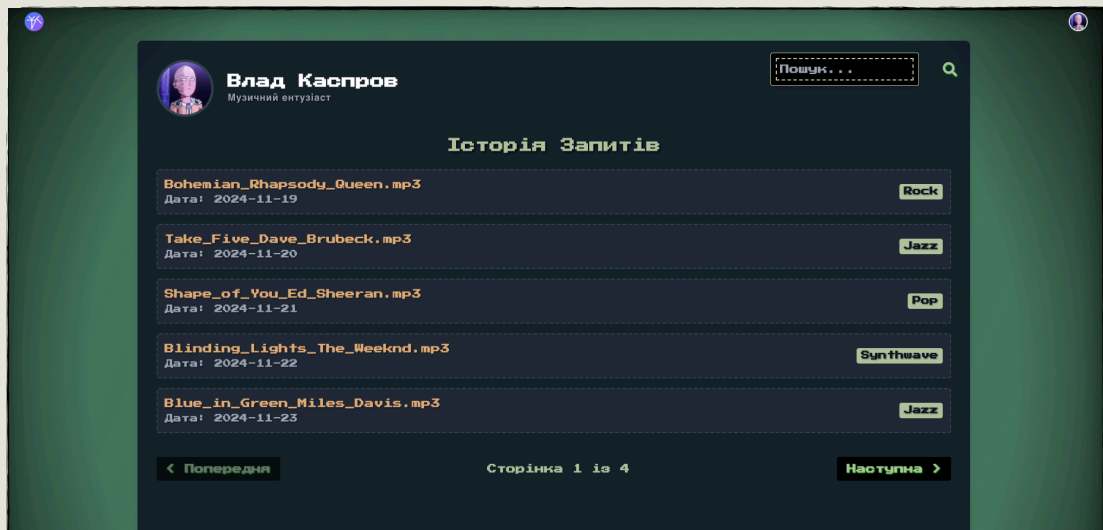


Рисунок Г.13 – Сторінка користувача веб-застосунку

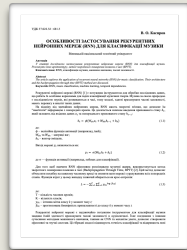
Сторінка авторизації



Рисунок Г.14 – Сторінка авторизації веб-застосунку

Апробація результатів роботи

Результати роботи було представлено на конференції "Молодь в науці: дослідження, проблеми, перспективи (МН-2025)"



Публікації

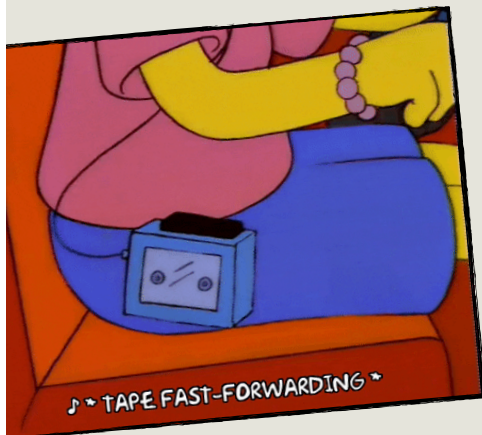


Каспров В.О. Особливості застосування рекурентних нейронних мереж (RNN) для класифікації музики Молодь в науці: дослідження, проблеми, перспективи (МН-2025)

Каспров В.О. Особливості застосування механізму уваги (Attention) у класифікації музичних творів Молодь в науці: дослідження, проблеми, перспективи (МН-2025)

Рисунок Г.15 – Апробація результатів роботи

ВИСНОВКИ



У процесі виконання магістерської кваліфікаційної роботи було розроблено ефективну систему класифікації музичних творів за жанрами на основі гібридної моделі глибинного навчання, що поєднує CNN, RNN та механізм уваги. Проведений аналіз предметної області виявив актуальність проблеми та недоліки існуючих рішень, а впровадження розробленої архітектури дозволило врахувати спектральні, темпоральні та семантичні особливості аудіосигналу, досягнувши приросту точності на 3% порівняно з наявними рішеннями. Отримані результати підтверджують доцільність застосування механізму уваги та глибинного навчання для підвищення точності, надійності та ефективності автоматизованої класифікації музичних творів.

Рисунок Г.16 – Висновки