# МАГІСТЕРСЬКА РОБОТА

МР.КІ – 06.00.00.000 ПЗ

Група $КІ_{М}$-24-1

Дутчак Владислав

2025

Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій

Кафедра комп'ютерних систем і мереж

**Дутчак Владислав Віталійович**
(прізвище, ім'я, по батькові)

УДК 004.4

# МАГІСТЕРСЬКА РОБОТА

## Розробка програмної системи автоматичної класифікації адіоданих на основі згорткової нейромережі

**Комп'ютерна інженерія**
**(назва освітньої програми)**

**123 – Комп'ютерна інженерія**
(шифр і назва спеціальності)

/ В. В. Дутчак /
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник – Гарасимів Віра Михайлівна, к.т.н., доцент

**Допущено до захисту**
Завідувач кафедри

д-р.т.н., проф. /С.І. Мельничук/
(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

доцент /Д. Р. Кропивницький/
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025 рік

**Івано-Франківський національний технічний університет нафти і газу**

Факультет *Інформаційних технологій*

Кафедра *Комп'ютерних систем і мереж*

Освітній рівень *магістр*

Спеціальність *123 – Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ**:**

Зав. кафедрою КСМ

проф. С. І. Мельничук

“ 05 ” грудня 2025 р.

# ЗАВДАННЯ

## НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТОВІ

*Дутчаку Владиславу Віталійовичу*

(прізвище, ім’я, по-батькові)

**1. Тема магістерської роботи** *Розробка програмної системи автоматичної класифікації аудіоданих на основі згорткової нейромережі*

**Керівник проекту** *доц., к.т.н. Гарасимів В. М.*

затверджені наказом вищого навчального закладу від « 05» *грудня 2025 року № 754/7.*

**Термін здачі студентом закінченої роботи** *10 грудня 2025 р*

**3. Вихідні дані до проекту (роботи)** *матеріали науково-дослідної практики*

**4. Зміст розрахунково - пояснювальної записки (перелік питань, що їх належить розробити)**

*Огляд предметної області.*

*Проектування системи класифікації аудіоданих на основі CNN.*

*Реалізація програмної системи та результати експериментів.*

**5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)**

**6. Консультанти по магістерській роботі, із зазначенням розділів роботи, що стосуються їх**

| Розділ | Консультант | Підпис, дата |
|---|---|---|
| **Нормоконтроль** | **Мойсеєнко О. В.** | |
| | | |

**7. Дата видачі завдання –** *12 березня 2025*.

# КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів магістерської роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | *Аналіз літератури пов'язаних з обраною темою* | *12.03.25 – 15.06.25* | *Виконано* |
| 2 | *Огляд предметної області.* | *16.06.25 –8.08.25* | *Виконано* |
| 3 | *Проектування системи класифікації аудіоданих на основі CNN* | *9.08.25 – 18.09.25* | *Виконано* |
| 4 | *Реалізація програмної системи та результати експериментів* | *19.09.25 –20.11.25* | *Виконано* |
| 5 | *Оформлення роботи* | *21.11.25 – 10.12.25* | *Виконано* |

**Студент-магістр** _____
(підпис)

**Керівник роботи** _____
(підпис)

# АНОТАЦІЯ

Магістерська робота містить 89 сторінок, 22 ілюстрації, 5 таблиць, список використаних джерел із 25 найменувань, 1 додаток.

Магістерська робота присвячена розв'язанню актуальної задачі розробки програмної системи автоматичної класифікації аудіоданих із використанням методів глибокого навчання.

У роботі виконано аналіз предметної області аудіокласифікації, розглянуто основні типи задач та сучасні підходи до обробки звукових сигналів. Обґрунтовано доцільність застосування згорткових нейронних мереж як базового інструменту для автоматичного виділення інформативних ознак із часово-частотних представлень аудіосигналів. Особливу увагу приділено методам попередньої обробки аудіоданих та формуванню ознакового простору на основі мел-частотних кепстральних коефіцієнтів.

Розроблено архітектуру програмної системи, що реалізує повний конвеєр обробки аудіо: від завантаження та нормалізації сигналу до інференсу нейромережевої моделі й формування класифікаційного рішення. Реалізацію системи виконано з використанням сучасних програмних засобів і бібліотек для цифрової обробки сигналів та машинного навчання. Проведено експериментальні дослідження ефективності розробленої системи на тестових наборах аудіоданих, отримано кількісні показники точності класифікації та проаналізовано результати роботи моделі.

Практичне значення роботи полягає у можливості застосування розробленої програмної системи для задач автоматичного аналізу та класифікації аудіосигналів у різних прикладних сферах, зокрема в системах розпізнавання звуків, музичному тегуванні, акустичному моніторингу та інтелектуальних користувацьких застосунках.

Ключові слова: аудіодані, класифікація, згорткова нейронна мережа, MFCC, глибоке навчання, обробка сигналів, програмна система.

## SUMMARY

The master's thesis comprises 89 pages, 22 figures, 5 tables, a list of references containing 25 sources, and 1 appendix.

The master's thesis is devoted to solving a relevant problem of developing a software system for the automatic classification of audio data using deep learning methods. The study provides an analysis of the audio classification domain, examines the main types of tasks, and reviews modern approaches to audio signal processing. The applicability of convolutional neural networks as a core tool for the automatic extraction of informative features from time–frequency representations of audio signals is substantiated. Particular attention is paid to audio data preprocessing methods and to the formation of the feature space based on Mel-frequency cepstral coefficients.

The architecture of a software system implementing a complete audio processing pipeline is developed, covering all stages from signal loading and normalization to neural network model inference and the formation of a classification decision. The system is implemented using modern software tools and libraries for digital signal processing and machine learning. Experimental studies of the developed system's effectiveness are conducted on test audio datasets; quantitative classification accuracy metrics are obtained, and the model performance is analyzed.

The practical significance of the work lies in the possibility of applying the developed software system to tasks of automatic analysis and classification of audio signals in various application domains, including sound recognition systems, music tagging, acoustic monitoring, and intelligent user applications.



Keywords: audio data, classification, convolutional neural network, MFCC, deep learning, signal processing, software system.

# ЗМІСТ

# ВСТУП

**Актуальність теми.** У сучасному цифровому світі обсяг мультимедійних даних, зокрема аудіоінформації, стрімко зростає. Важливими є завдання автоматичного розпізнавання, сегментації та класифікації аудіосигналів, що мають широке застосування у таких галузях як безпека (виявлення звуків сирени, вибухів), охорона довкілля (моніторинг біозвуків), розпізнавання мовлення, автоматичне тегування музики, інтелектуальні системи підтримки користувача. Зростає попит на системи, здатні ефективно та точно класифікувати аудіосигнали в реальному часі.

Низка провідних компаній, таких як Google, Meta, Amazon і Spotify, активно впроваджують технології автоматичної обробки звуку у свої продукти. Наприклад, Google у проєкті AudioSet створив великий датасет із позначених аудіоподій, що стимулює розвиток досліджень у цій галузі. Подібні системи лежать в основі таких сервісів, як Shazam (ідентифікація музики), Alexa та Siri (розпізнавання команд), YAMNet і VGGish (предтреновані моделі для аудіокласифікації).

Особливу ефективність у задачах класифікації аудіо продемонстрували згорткові нейронні мережі (Convolutional Neural Networks, CNN), здатні витягати та аналізувати просторово-часові закономірності у спектрограмах чи MFCC-ознаках сигналів. Це робить тему даної магістерської роботи актуальною з точки зору як наукової новизни, так і практичної корисності.

**Мета і завдання дослідження.**

Метою магістерської роботи є розробка програмної системи, здатної автоматично класифікувати аудіодані з використанням згорткової нейронної мережі, орієнтуючись на ефективність, точність та універсальність застосування.

Для досягнення цієї мети необхідно вирішити такі завдання:

1 Проаналізувати сучасні підходи до автоматичної класифікації аудіосигналів.

2 Визначити та обґрунтувати оптимальний набір ознак для представлення аудіоінформації (MFCC, спектрограма тощо).

3 Розробити структуру згорткової нейронної мережі для розв’язання задачі багатокласової класифікації.

4 Реалізувати програмну систему класифікації із графічним інтерфейсом.

5 Провести експериментальні дослідження точності класифікації та оцінити ефективність реалізованої системи.

**Об’єктом дослідження** є процес автоматизованої класифікації аудіосигналів у технічних системах розпізнавання.

**Предметом дослідження** є методи та засоби класифікації аудіоданих на основі ознак, витягнутих із сигналів, із застосуванням згорткових нейронних мереж.

**Методи дослідження.** У роботі використовуються методи цифрової обробки сигналів (включаючи перетворення Фур’є та мел-частотні кепстральні коефіцієнти — MFCC), методи машинного навчання та глибокого навчання, зокрема згорткові нейронні мережі (CNN), а також програмні інструменти Python, Keras, TensorFlow для реалізації й навчання моделей.

**Практичне значення.** Розроблену систему можна застосовувати у практиці побудови розумних голосових помічників, систем виявлення подій у міському середовищі, автоматизованого моніторингу стану обладнання за звуком, а також у медіа-індустрії — для класифікації музики, звуків природи або шумів. Реалізація зручного інтерфейсу користувача сприяє інтеграції системи у реальні застосунки.

**Структура і обсяг магістерської роботи.** Робота написана обсягом 89 сторінок і містить 22 ілюстрації, 5 таблиць та 25 джерел.

# 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Предметна область та актуальні задачі аудіокласифікації

Класифікація аудіоданих – це процес автоматичного віднесення фрагмента звукового сигналу до одного з наперед визначених класів на основі його змістовних характеристик [1].

На відміну від розпізнавання мовлення (де потрібно перетворити мовний сигнал у текст) або ідентифікації конкретного звуку (наприклад, знайти ту саму пісню, як робить Shazam), під задачами класифікації найчастіше розуміють визначення типу чи категорії звуку. Прикладами таких задач є:

1. Класифікація музики за жанрами. Метою є автоматично визначити жанр або стиль музичного твору (рок, джаз, класика, поп тощо) за його аудіозаписом. Це класична проблема, над якою працювали ще з початку 2000-х. Піонерською вважається робота Джорджа Цанетакіса (Tzanetakis, 2002), який створив датасет GTZAN і застосував традиційні ознаки (включно з MFCC) та класифікатори для розпізнавання жанрів. Класифікація за жанрами корисна для впорядкування музичних бібліотек, рекомендацій та музикознавчих досліджень.

2. Розпізнавання акустичних сцен та подій. Тут аудіоаналіз спрямований на визначення, що відбувається у звуковому середовищі або яке джерело звуку присутнє. Наприклад, задача класифікації звукових подій може визначати, чи запис містить звук двигуна автомобіля, гавкіт собаки, шум дощу, людську мову тощо. Ця сфера активно розвивається в рамках конкурсу DCASE (Detection and Classification of Acoustic Scenes and Events), де щорічно пропонуються датасети міських шумів, побутових звуків тощо. Практичне значення – від смарт-моніторингу міст (шумове забруднення, сигнали небезпеки) до побудови систем “розумного дому”, що реагують на певні звуки (розбиття скла, плач дитини) [2].

3. Класифікація мовлення та мов. Приклад – визначення, якою мовою говорить людина у записі (англійська, українська, китайська тощо). Інший

приклад – класифікація емоцій за інтонацією мовлення (емоційна забарвленість голосу). Такі системи можуть бути корисні для кол-центрів (визначення незадоволеного клієнта по голосу) або мультимедійних додатків.

4. Ідентифікація мовця та біометрична класифікація. Хоч це дещо інша задача, але споріднена: визначити, хто говорить (ідентифікація особи за голосом) або навіть деякі атрибути – стать мовця, вікова група. Ці задачі теж часто вирішуються шляхом класифікації аудіохарактеристик [3].

Загальною особливістю всіх цих задач є те, що аудіосигнал повинен бути перетворений у набір характеристик (ознак), за якими потім алгоритм ухвалить рішення про клас. Аудіо – це часовий сигнал, тому простий набір сирих відліків не є безпосередньо зручним для класифікації (через велику розмірність і залежність від зсуву в часі). Тому першим етапом є особлива форма подання аудіо, яка б відображала його частотний склад і перцептивні особливості. Найпоширенішими представленнями є спектрограма, мел-спектрограма або набір статистичних ознак, що характеризують спектр [4].

Спектрограма – це зображення, що показує, як розподіл енергії по частотах змінюється з часом. Вона будується шляхом розбиття сигналу на короткі інтервали (фрейми), обчислення спектру (наприклад, швидким перетворенням Фур'є) для кожного фрейму і відображення спектральної щільності кольором або яскравістю. Мел-спектрограма – варіант спектрограми, де вісь частот перетворена до нелінійної мел-шкали, яка моделює особливості людського слуху: на низьких частотах розрізнення тонких відмінностей краще, ніж на високих [5].

Мел-шкала визначається приблизно формулою:

$$\boldsymbol{m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right)}, \tag{1.1}$$

де $f$ – частота в герцах, $m$ – відповідна частота в мелах. Така шкала стискає високочастотну область, зменшуючи роздільність там, де людське вухо менш чутливе до різниці частот.

Мел-кепстральні коефіцієнти (MFCC) – це компактний набір ознак, який широко застосовується в обробці мови і звуку. MFCC є результатом обчислення спектра на мел-шкалі та подальшого переходу в кепстральний (частотно-часовий) домен. По суті, MFCC репрезентують тімбральну інформацію звуку, тобто огинаючу спектру (спектральний контур), яка пов'язана зі тембром джерела [6]

Алгоритм обчислення MFCC можна описати так: сигнал проходить через ряд кроків (рис. 1.1) [7]:

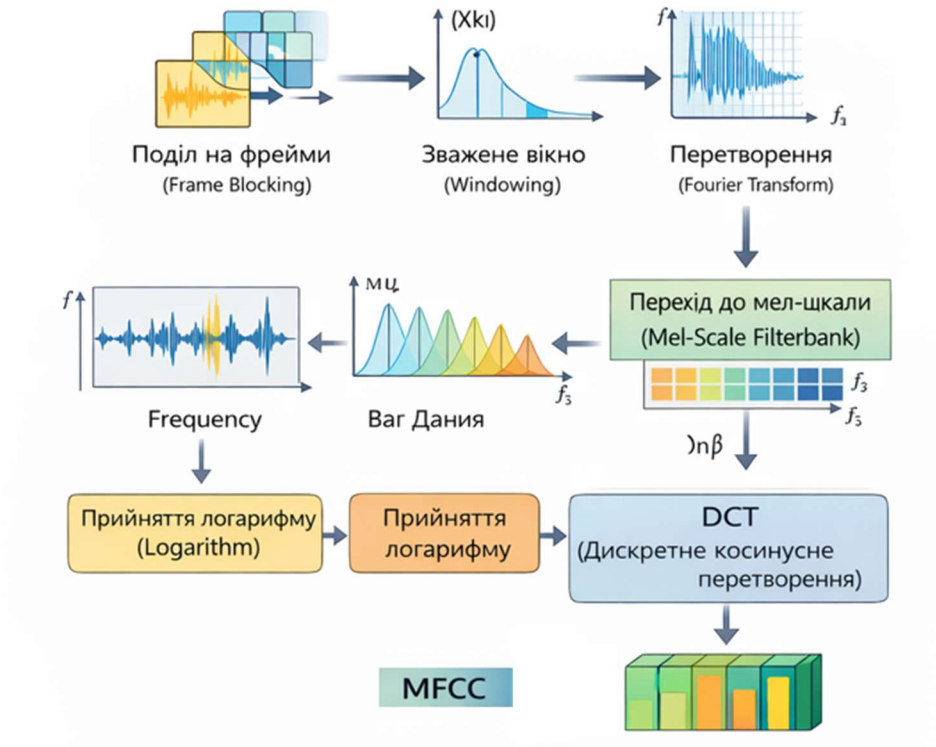


Рисунок 1.1 – Алгоритм обчислення MFCC

1. Попередня обробка: часто застосовують преемфазу – підсилення високих частот, щоб компенсувати спектральний спад мовних сигналів і покращити співвідношення сигнал/шум на високих частотах.

2. Потім сигнал ділять на короткі фрейми (наприклад, 25 мс із кроком 10 мс) з накладанням вікна (зазвичай вікно Гаммінга) для згладжування країв відрізка і зменшення побічних пелюсток при спектральному аналізі.

3. Перетворення Фур'є: для кожного фрейму обчислюється дискретне перетворення Фур'є (DFT), щоб отримати амплітудний спектр $\boldsymbol{X_k}$. Перетворення Фур'є забезпечує перехід у частотну область, де видно, які частоти присутні у сигналі. Отриманий спектр зазвичай перетворюють у спектр потужності $\boldsymbol{P[k] = |X_k|^2}$.

4. Мел-фільтрація: спектр потужності проходить через набір фільтрів з трикутною частотною характеристикою, рівномірно розташованих по мел-шкалі. Кожен фільтр $\boldsymbol{H_j(f)}$ виділяє енергію в певному частотному діапазоні. Результат – значення енергії в смузі $\boldsymbol{E_j = \sum_k P[k] \cdot H_j(k)}$. Фільтрів беруть певну кількість (типово 20–40), що визначає число подальших коефіцієнтів. Перетворення до мелового спектру відображає особливості людського сприйняття: дрібні відмінності в низьких частотах зберігаються, а у високих – згладжуються.

5. Логарифмування: до виходу кожного фільтра застосовується логарифм: $\boldsymbol{L_j = \ln E_j}$. Це переводить мультиплікативні зміни у сигналі (наприклад, зміни амплітуди) в адитивну форму і наближає розподіл значень до нормального. Логарифм енергії також певним чином моделює нелінійність гучності в людському слуху (рівномірне збільшення гучності відповідає експоненційному зростанню енергії сигналу).

6. Перехід у кепстральний домен: виконується дискретне косинусне перетворення (DCT) над послідовністю $\boldsymbol{L_j}$. Коефіцієнти DCT обчислюються за формулою:

$$\boldsymbol{c_m = \sum_{j=1}^{M} L_j \cos!\left[\frac{\pi m}{M}\left(j - \frac{1}{2}\right)\right]}, \quad (1.2)$$

де $M$– число фільтрів (розмірність мел-спектру), $\boldsymbol{m = 0, 1, \dots, M-1}$. В результаті отримуємо набір кепстральних коефіцієнтів $\boldsymbol{c_m}$. Перший з них ($\boldsymbol{c_0}$) відповідає середній енергії в спектрі, його часто опускають або розглядають окремо. Наступні коефіцієнти $\boldsymbol{c_1, c_2,}$ описують основну форму спектральної огинаючої (тембр): $\boldsymbol{c_1}$ пов'язаний із загальним нахилом спектра (і корелює зі

спектральним центроїдом), вищі коефіцієнти додають деталі (впливають на представлення висоти тону, формант тощо). Зазвичай використовують перші 12–13 MFCC, оскільки вони вміщують основну інформацію про форму спектра, тоді як вищі коефіцієнти описують дрібні деталі і часто можуть бути відкинуті як шум або малозначуща варіація.

Таким чином, вихідним результатом є вектор з $m$ коефіцієнтів MFCC для кожного фрейму. Часто також додають перші та другі різниці (дельта-коефіцієнти), щоб врахувати динаміку змін ознак у часі. Комплект з 13 базових MFCC + 13 дельта + 13 дельта-дельта дає 39-розмірний вектор ознак на фрейм – такий формат історично використовувався в системах розпізнавання мовлення.

MFCC надзвичайно популярні в силу їх ефективності: вони компактно описують спектр, мають певну стійкість до шумів і узагальнюють ключові особливості звуку, ігноруючи несуттєві варіації. Як зазначається в оглядах, MFCC стали стандартом у більшості аудіо-доменів, успішно застосовані і для музики (оцінка тембру) та інших звуків. Немає строгого теоретичного обґрунтування, чому саме мел-шкала краща за інші можливі шкали (існують варіації, де замість мел використовують шкалу Барк або ERB) – проте історично саме MFCC показали себе добре і закріпилися як базовий підхід.

**Інші ознаки.** Окрім MFCC, для аудіо застосовуються й інші ознаки: хрома-фічі (характеризують енергетику по музичних нотах незалежно від октави, важливі для визначення тональності музики), Zero-Crossing Rate (частота переходів сигналу через нуль – часто використовується для відмінності шуму/тону), Spectral Roll-off (частота, нижче якої знаходиться певний відсоток енергії спектра), спектральна площинність та багато інших. У минулому дослідники формували великі набори ручних ознак і намагалися вибрати з них найінформативніші для певної задачі. Однак зараз домінує підхід, при якому нейронна мережа сама навчається вилучати ознаки з більш сирого представлення (наприклад, зі спектрограми чи навіть безпосередньо з сирого аудіосигналу) [8].

## 1.2 Методи та системи автоматичної класифікації аудіо: від класичних до нейронних

Як зазначалося, класичний підхід до класифікації аудіо побудований на зв'язці «ручні ознаки + стандартний класифікатор». У випадку музичної класифікації за жанрами типовий конвеєр виглядав так [9]:

1. Вибір набору ознак. На основі експертних знань про звук обиралися характеристики, які можуть найкраще розрізняти потрібні класи. Для музичних жанрів, наприклад, бралися середні та дисперсійні значення MFCC по всьому треку (відображають тембр), ознаки ритму (темп, періодичність ударів), наявність вокалу (через спектральні ознаки), тональні характеристики (хрома-фічі для гармонійності) тощо.
2. Виділення ознак з аудіозаписів. Кожен файл у наборі даних оброблявся для обчислення вибраного набору ознак. Часто аудіо потребувало нормалізації гучності, обрізання або приведення до єдиного формату (моно, 16 кГц), розбиття на короткі фрагменти і усереднення ознак, якщо потрібно.
3. Вибір алгоритму класифікації. Популярними були методи статистичного навчання: Gaussian Mixture Models (GMM) – ймовірнісні моделі, що оцінюють розподіл ознак кожного класу (наприклад, для музики моделюють розподіл MFCC для кожного жанру); Support Vector Machine (SVM) – пошук гіперплощини, що розділяє класи з максимальним відступом у просторі ознак; k-Nearest Neighbors (kNN) – віднесення до класу більшості найближчих сусідів у просторі ознак; Random Forest та інші. Для послідовних даних (наприклад, звуки, які розвиваються в часі, мовні фонеми) застосовувалися приховані моделі Маркова (HMM), які добре працювали разом з MFCC у розпізнаванні мовлення.
4. Вибір таксономії класів. Для музики, наприклад, необхідно визначити список жанрів і їх ієрархію, оскільки існують різні рівні узагальнення жанрів. Для звукових подій – вирішити, які категорії цікаві (наприклад, для побутових звуків одна таксономія, для природних – інша) [10].

5. Навчання і класифікація. Класифікатор тренувався на підготовлених ознаках для тренувальної вибірки, потім перевірявся на тестових даних, після чого використовувався для прогнозування класу нових аудіо.

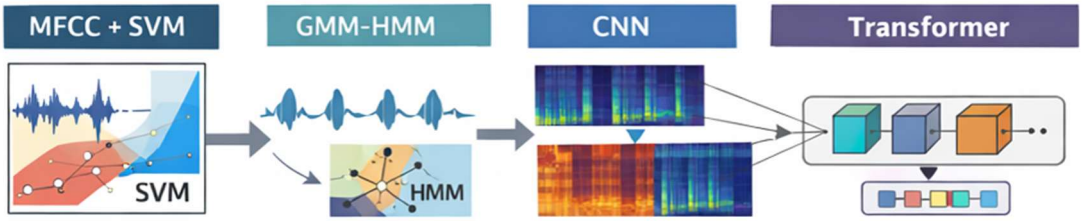


Рисунок 1.2 – Методи класифікації аудіо

За такого підходу критично важливим був крок 1 – інженерія ознак. Від якості та релевантності обраних ознак сильно залежали результати. Наприклад, у визначенні жанрів музики темброві ознаки (MFCC) виявилися дуже корисними, адже різні жанри мають різний інструментальний склад і характер звучання, що відображається в спектральній формі сигналу. З іншого боку, для розпізнавання окремих звукових подій (скажімо, відрізнити гавкіт від плачу дитини) тембр теж важливий, але додатково потрібні ознаки форми сигналу в часі. Через ці складнощі traditional підхід вимагав багато експериментів і експертизи для кожної нової задачі [11].

Методи класифікації аудіосигналів еволюціонували від статистичних моделей і простих фрейм-ознак до складних глибоких нейромереж, здатних навчатися на великих акустичних масивах даних. У таблиці 1.1 подано узагальнене порівняння основних груп підходів [12].

Глибокі нейронні мережі спростили цей процес, автоматизувавши витяг ознак. Згорткові мережі, зокрема, добре відомі здатністю утворювати ієрархію ознак: нижчі шари реагують на прості локальні патерни, вищі – на більш складні та інваріантні характеристики. У випадку аудіо найпопулярнішим рішенням стало представлення аудіосигналу у вигляді «зображення» – часово-частотної карти (спектрограми або мел-спектрограми) – і застосування 2-вимірної CNN,

аналогічно до обробки зображень. Такий підхід використовували Піцзак (2015) та багато інших дослідників з середини 2010-х. Згорткові нейромережі для аудіо показали суттєве покращення в порівнянні з класичними алгоритмами на множині задач. Дослідники з університету Карнегі–Меллон (CMU) і Google в 2014 р. продемонстрували успіх CNN у розпізнаванні мовлення, використовуючи вхідні ознаки у вигляді лог-мел-спектрограм. У 2017 р. Google представив результати на великому наборі даних AudioSet: різні архітектури CNN (в тому числі подібні до VGG і ResNet) досягли високої точності у класифікації 527 аудіо-категорій, заклавши основу для моделей на зразок VGGish [13-15].

Таблиця 1.1 – Порівняння методів класифікації аудіо

| **Метод** | **Тип ознак** | **Підхід** | **Переваги** | **Недоліки** |
|---|---|---|---|---|
| MFCC + SVM | MFCC, LPC | Класичний | Простота, швидке навчання | Низька точність, чутливість до шуму |
| HMM (GMM-HMM) | MFCC | Статистичний | Гарна робота для мовлення | Складність налаштування, стаціонарність |
| Decision Trees | Числові ознаки | Евристичний | Інтерпретованість | Обмежена узагальнювальність |
| CNN (2D) | Спектрограма | Глибоке навчання | Висока точність, простота структури | Потребує багато даних |
| CRNN | Спектрограма | CNN + RNN | Облік контексту, сильні результати | Складність архітектури |
| Transformers (AST) | Патчі спектрограм | Attention-механізм | Глобальний контекст, гнучкість | Великі обчислювальні витрати |

В Україні та світі протягом останніх років з’являлися роботи, де CNN адаптувалися для різних аудіозадач. Наприклад, Іванов О. В. (2019) розглянув застосування таких відомих архітектур, як AlexNet і VGG, для класифікації звукових подій із набору UrbanSound8K (10 класів звуків). Було встановлено, що спеціально підібрані CNN можуть успішно розрізняти навіть складні звуки довкілля. Інший напрям – рекурентні нейронні мережі (RNN) та трансформери, які враховують послідовну природу аудіо. Їх доцільно застосовувати в задачах,

де важлива довготривала структура (наприклад, аналіз музичної композиції протягом хвилин). У нашій роботі основна увага приділяється саме CNN, оскільки вони добре виявляють локальні часово-частотні шаблони звуку (наприклад, гармоніки, форманти, удари) і при цьому менш вибагливі до обсягу даних, ніж трансформери. Також CNN відносно ефективні обчислювально, що дозволяє використовувати їх у реальному часі та на пристроях з обмеженими ресурсами (наприклад, мобільних телефонах для задач типу “OK Google”, “Hey Siri”) [16-18].

Згорткові нейронні мережі (***CNN***) були адаптовані для задач класифікації аудіосигналів шляхом подання спектрограм у вигляді двовимірних зображень. Наприклад, модель ***VGGish***, запропонована Hershey et al. (2017), базується на спрощеній архітектурі ***VGG*** і продемонструвала високі результати на великомасштабному аудіодатасеті ***AudioSet*** **[19]**.

Базовий згортковий шар ***CNN*** може бути формалізований як операція згортки:

$$\boldsymbol{Y_{i,j}^{(l)} = f!\left(\sum_{m,n} X_{i+m,j+n}^{(l-1)} \cdot K_{m,n}^{(l)} + b^{(l)}\right)}, \quad (1.3)$$

де $\boldsymbol{X^{(l-1)}}$ — вхідне тензорне представлення з попереднього шару, $\boldsymbol{K_{m,n}^{(l)}}$ — ядро згортки $\boldsymbol{l}$-го шару,$\boldsymbol{b^{(l)}}$— зсув (bias), а $\boldsymbol{f(\cdot)}$ — нелінійна активаційна функція (наприклад, ***ReLU***).

У такій інтерпретації кожен згортковий фільтр навчається реагувати на певні локальні патерни у спектрограмі, наприклад вертикальні структури, що відповідають гармонікам, або горизонтальні структури, пов’язані з шумовими компонентами. Завдяки спільному використанню ваг та локальній рецептивній області ***CNN*** демонструють високу ефективність і стійкість до зсувів у часі та частоті, що робить їх особливо придатними для аналізу аудіосигналів.

**Огляд існуючих систем та рішень.** Окрім згаданих Shazam, Spotify, YAMNet, варто відзначити інші помітні рішення:

– Система SoundHound – подібно до Shazam, розпізнає музику, але також здатна визначати проспівавши мелодію. Використовує складні аудіо-фінгерпринти та елементи штучного інтелекту [19];

– Google Assistant та Amazon Alexa – голосові помічники, які вміють не лише розпізнавати слова, а й виконувати базову класифікацію аудіо-команд (наприклад, відрізнити команду від фону). В їх основі поєднання згорткових мереж (для вилучення ознак) та рекурентних/трансформерних моделей [20, 21];

– YAMNet (Google, 2019) – як вже згадано, нейромережевий аудіо-класифікатор, попередньо навчений на величезному наборі YouTube-видео AudioSet. Він став доступним широкому загалу через TensorFlow Hub з готовими вагами, і фактично є універсальним екстрактором ознак: можна подати будь-який звук (наприклад, шум вулиці), і YAMNet видасть ймовірності понад 500 категорій. Модель побудована на полегшеній архітектурі MobileNet (глибинні роздільні згортки), що забезпечує швидкість і невеликий розмір [22];

– VGGish (Google, 2017) – модель, що генерує 128-розмірний ембедінг (вектор ознак) з ~1 секунди аудіо. Архітектура мережі нагадує VGG16, але пристосована до входу 96 × 64 (час × частота) мел-спектрограми. VGGish тренувався на YouTube даних (8 млн відео) для задачі авто-тегування аудіо. На відміну від YAMNet, VGGish не намагається безпосередньо класифікувати звук у конкретний клас, а слугує універсальним перетворювачем аудіо у «смисловий простір» ознак, придатний для використання іншими моделями. У практиці VGGish використовується, наприклад, для детекції емоцій або подій: спочатку отримують ембедінги, а потім навчають простішу модель (наприклад, SVM або невелику нейронну мережу) класифікувати ці ембедінги на потрібні класи [23];

– OpenL3 (2018) – модель для аудіо-ембеддінгів на основі архітектури AudioSet (L3 – “Look, Listen and Learn”), яка також може слугувати фічер-екстрактором [24];

– Kaggle моделі та інше. На платформах змагань (Kaggle тощо) регулярно з’являються прикладні рішення для аудіокласифікації – від розпізнавання звуків

природи до класифікації жанрів. Наприклад, на Kaggle відкрито доступні ноутбуки з прикладами класифікації набору ESC-50 за допомогою CNN, що досягають ~85% точності (для порівняння: традиційний підхід MFCC+SVM давав ~70% на ESC-50) [25].

Таким чином, огляд показує, що на сьогодні нейронні мережі, зокрема згорткові, є домінуючим підходом у задачах класифікації аудіосигналів завдяки їх високій якості й можливості автоматично виділяти ознаки. Комерційні системи (Spotify, Shazam) теж використовують елементи такого підходу, комбінуючи їх зі спеціалізованими алгоритмами (фінгерпринти, колаборативна фільтрація тощо) для вирішення своїх конкретних завдань.

Застосування систем класифікації аудіо:

1. Розпізнавання мовлення. Системи, як-от Whisper (OpenAI), Kaldi, Google Speech API, використовують CNN та трансформери для покращення точності розпізнавання, зокрема в шумних середовищах.

2. Акустичний моніторинг довкілля. Проєкти, як CitySounds або SONYC, застосовують класифікацію звуків міста (авто, сирени, індустріальний шум), з метою виявлення порушень шумових норм або небезпеки.

3. Безпека. Розпізнавання пострілів, розбитого скла, криків — критично важливе для інтелектуального відеоспостереження. Моделі CNN дозволяють знизити час виявлення до сотих долей секунди.

4. Музичне тегування та рекомендації. Deep-learning підходи автоматизують класифікацію музичних треків за жанрами, виконавцями, емоціями, що формує основу рекомендаційних систем (Spotify, YouTube Music).

### 1.3. Постановка задачі дослідження

На основі аналізу предметної області сформулюємо конкретну задачу, яка вирішується у межах даної магістерської роботи.

Мета роботи – розробити прототип програмної системи, що здійснює автоматичну класифікацію аудіоданих на основі CNN. Для демонстрації можливостей системи обрано конкретний сценарій: класифікація музичних фрагментів за жанрами. Це одна з типових задач аудіоаналізу, яка достатньо складна (жанри можуть мати схожі риси) і водночас має практичне значення (наприклад, для автоматичного тегування музики). Таким чином, система на вході отримує аудіофайл із музичним твором і повертає передбачений жанр цього твору. При бажанні, розроблене рішення може бути розширене і на інші види аудіокласифікації – зокрема, шляхом перенавчання моделі на інших датасетах (звуки довкілля, мова тощо).

Завдання, які необхідно вирішити для досягнення мети:

– Розробити загальну архітектуру програмної системи аудіокласифікації. Вона має включати модуль завантаження та попередньої обробки аудіо, модуль екстракції ознак (MFCC або спектрограми), модуль нейронної мережі для класифікації та інтерфейс користувача для взаємодії із системою.

– Обрати та обґрунтувати набір даних для навчання і тестування моделі. Для задачі класифікації жанрів музики доцільно використати відкриті набори, такі як GTZAN (1000 треків, 10 жанрів) або FMA (Free Music Archive dataset). У роботі планується використання набору GTZAN як прикладного (він широко застосовується у наукових працях для цієї задачі).

– Виконати попередню обробку аудіосигналів: нормалізацію гучності, приведення до єдиної довжини або вибірка сегментів (наприклад, використовувати перші 30 секунд кожного треку), конвертація до потрібної частоти.

– Спроектувати архітектуру згорткової нейронної мережі для класифікації за жанрами. Враховуючи, що вхідні дані можуть бути представлені як «зображення» розміру (n_mfcc × T), де n_mfcc – число коефіцієнтів, T – кількість фреймів у аудіофрагменті, мережа повинна містити згорткові шари

(2D-конволюції) для автоматичного виділення ознак високого рівня з цих карт ознак.

– Реалізувати програмно всю систему: написати код для завантаження даних, обчислення ознак (з використанням бібліотек, таких як Librosa для Python), побудови та навчання моделі (на базі TensorFlow/Keras або PyTorch), а також код для інференсу – щоб можна було подати новий аудіофайл і отримати передбачення класу.

– Створити простий інтерфейс користувача (настільний додаток або веб-інтерфейс) для демонстрації роботи системи.

– Провести тестування та оцінку точності моделі. Для цього на тестовому піднаборі (або перехресною перевіркою) отримати метрики класифікації: точність (accuracy), матрицю змішування, показники Precision/Recall/F1 для кожного класу. Порівняти отриманий результат з відомими в літературі (для GTZAN відомо, що ~80% accuracy досягається сучасними моделями, а межа ~85% є дуже доброю, враховуючи деяку зашумленість самого датасету GTZAN).

В даному розділі виконано огляд предметної області аудіокласифікації та розглянуто основні типи задач, що виникають під час автоматичного аналізу звукових сигналів, зокрема класифікацію музики, мовлення, акустичних сцен і подій. Проаналізовано класичні підходи на основі ручної інженерії ознак і статистичних класифікаторів, а також сучасні методи глибокого навчання, зокрема згорткові нейронні мережі, які забезпечують автоматичне виділення інформативних ознак і демонструють вищу точність у порівнянні з традиційними рішеннями.

На підставі огляду існуючих методів і систем сформульовано постановку задачі дослідження, що полягає у розробці прототипу програмної системи автоматичної класифікації аудіоданих на основі CNN з прикладною орієнтацією на класифікацію музичних фрагментів за жанрами. Визначено мету роботи, основні завдання та вимоги до архітектури системи, наборів даних, процесів попередньої обробки, навчання й оцінювання моделі.

# 2 ПРОЕКТУВАННЯ СИСТЕМИ КЛАСИФІКАЦІЇ АУДІОДАНИХ

У цьому розділі розробляється архітектура програмної системи та алгоритми обробки аудіоданих і класифікації. Відповідно до вимог, основні компоненти системи: модуль підготовки аудіосигналу та виділення ознак, згорткова нейронна мережа як ядро класифікації, а також компоненти, що забезпечують інтеграцію моделі в готовий застосунок (завантаження моделі, отримання передбачень, інтерфейс для користувача). Також в розділі подано математичні основи реалізованих алгоритмів, структурні схеми та UML-діаграми, що відображають дизайн рішення.

## 2.1. Аналіз та формалізація функціональних вимог до системи

Проєктування програмної системи автоматичної класифікації аудіоданих потребує чіткого визначення її функціональних можливостей, які безпосередньо випливають із поставленої задачі та аналізу предметної області, проведеного у першому розділі. Формалізація вимог є необхідною умовою для побудови коректної архітектури, вибору алгоритмів обробки сигналів та подальшої реалізації програмного забезпечення.

Розроблювана система повинна забезпечувати повний цикл обробки аудіосигналу — від моменту надходження вхідного файлу до отримання результату класифікації у вигляді ймовірнісного або категоріального рішення. З огляду на це, функціональні вимоги до системи доцільно структурувати відповідно до основних етапів обробки аудіоданих.

Вимоги до модуля введення та керування даними. Система повинна забезпечувати можливість завантаження аудіофайлів користувачем у поширених форматах (WAV, MP3, FLAC). При цьому необхідно передбачити перевірку коректності вхідних даних, зокрема контроль частоти дискретизації, кількості каналів та тривалості сигналу. У випадку невідповідності параметрів вхідного

аудіо встановленим вимогам система повинна виконувати автоматичне приведення сигналу до стандартного вигляду або повідомляти користувача про помилку.

Вимоги до модуля попередньої обробки сигналів. Система повинна реалізовувати алгоритми попередньої обробки аудіосигналів, необхідні для стабільної роботи класифікатора. До таких алгоритмів належать:

– нормалізація амплітуди сигналу;
– приведення частоти дискретизації до фіксованого значення;
– сегментація сигналу на перекривні фрейми фіксованої довжини;
– застосування віконної функції для зменшення спектральних спотворень.

Цей етап є критично важливим, оскільки якість попередньої обробки безпосередньо впливає на інформативність подальших ознак.

Вимоги до модуля виділення ознак. Система повинна забезпечувати автоматичне виділення акустичних ознак із попередньо обробленого аудіосигналу. Основним типом ознак у межах даної роботи є мел-частотні кепстральні коефіцієнти (MFCC), які довели свою ефективність у задачах аудіокласифікації. Модуль виділення ознак повинен:

– формувати спектр сигналу за допомогою швидкого перетворення Фур’є;
– застосовувати банк мел-фільтрів;
– виконувати логарифмування спектральних енергій;
– здійснювати дискретне косинусне перетворення для отримання MFCC-векторів.

Отримані вектори ознак повинні мати фіксований розмір і бути придатними для подачі на вхід згорткової нейронної мережі.

Вимоги до модуля класифікації. Модуль класифікації повинен реалізовувати згорткову нейронну мережу, здатну обробляти двовимірні представлення аудіоознак (MFCC або спектрограми). Система повинна забезпечувати:

– завантаження попередньо навченої моделі;

– обчислення вихідних ймовірностей для кожного класу;

– вибір фінального класу на основі функції softmax.

Архітектура класифікатора повинна бути модульною, що дозволить у майбутньому змінювати кількість класів або адаптувати модель до інших типів аудіоданих.

Вимоги до модуля взаємодії з користувачем. Система повинна містити інтерфейс користувача, який забезпечує зручну взаємодію з програмою. Інтерфейс має надавати можливість:

– завантаження аудіофайлу;

– запуску процедури класифікації;

– відображення результатів у зрозумілому вигляді (назва класу, імовірність).

Загальні функціональні вимоги. Узагальнюючи наведене, розроблювана система повинна:

– працювати в автоматичному режимі без необхідності ручного налаштування параметрів;

– забезпечувати стабільну роботу при обробці різних типів аудіосигналів;

– бути розширюваною з точки зору алгоритмів і класів аудіо.

## 2.2 Алгоритмічне забезпечення програмної системи

Алгоритмічне забезпечення розроблюваної системи задає формалізований порядок виконання операцій від моменту отримання аудіофайлу до формування кінцевого класифікаційного рішення. На цьому рівні опису не обмежуються загальними міркуваннями про “передобробку” чи “виділення ознак”, а фіксують чітку послідовність кроків, що виконуються у визначеному порядку, із конкретизацією вхідних/вихідних даних для кожного етапу. Така фіксація важлива з двох причин: по-перше, вона забезпечує відтворюваність

експериментів (однаковий сигнал, однакові параметри — однаковий результат), по-друге, спрощує програмну реалізацію, оскільки дозволяє розділити рішення на автономні модулі (завантаження, нормалізація, ознаки, формування тензора, інференс, інтерпретація результату, логування).

У межах цієї роботи процес класифікації аудіоданих доцільно розглядати як конвеєр: послідовність взаємопов'язаних етапів, де вихід попереднього кроку є входом наступного. Типова структура такого конвеєра включає: завантаження аудіо та приведення параметрів, попередню обробку сигналу, виділення ознак (MFCC або мел-спектрограма), нормалізацію ознак, узгодження розмірностей і формування тензора для моделі, інференс CNN і, зрештою, перетворення виходу моделі у зрозумілий користувачеві результат (клас імовірностей або список топ-класів).

**Алгоритм загального конвеєра обробки аудіо та класифікації**

Нехай вхідним об'єктом є аудіофайл $f$, а виходом системи — прогнозований клас $\hat{y}$ та вектор імовірностей $p$. Загальну логіку функціонування можна подати як послідовність перетворень даних від сирого сигналу до рішення моделі:

$$f \to x(t) \to \tilde{x}(t) \to \Phi(\tilde{x}) \to X \to \mathrm{CNN}(X) \to p \to \hat{y},. \qquad (2.1)$$

Тут $x(t)$ — зчитаний (сирий) сигнал у часовій області $\tilde{x}(t)$ — сигнал після приведення параметрів і попередньої обробки; $\Phi(\cdot)$ — оператор формування ознак (MFCC або мел-спектрограма); $X$ — тензор, який подається на вхід CNN у погодженому форматі (наприклад, "каналізована" матриця ознак із фіксованою шириною по часу). Важливо підкреслити, що система реалізується так, щоб одна й та сама процедура передобробки й формування ознак застосовувалась і під час навчання, і під час використання (runtime). Це зменшує ризик "розриву" між тренувальним і реальним режимом роботи.


Рисунок 2.1 — Загальний алгоритм функціонування системи (конвеєр класифікації аудіо)

Алгоритм загального конвеєра класифікації аудіо:

1 Завантажити аудіофайл $f$ та зчитати аудіосигнал $x(t)$ разом із частотою дискретизації $f_s$.

2 Привести сигнал до єдиного стандарту: за потреби перетворити в моно, виконати ресемплінг до заданої частоти $f_s^*$ (наприклад, 16 кГц або 22{,}05 кГц), а також нормалізувати амплітуду, щоб зменшити вплив різного рівня запису.

3 Виконати сегментацію сигналу на фрейми фіксованої довжини $L$ із кроком $H$ (перекриття дозволяє стабілізувати спектральні оцінки та не "губити" короткі події).

4 Для кожного фрейму сформувати ознаки через оператор $\Phi(\cdot)$: або MFCC, або мел-спектрограму (вибір визначається дизайном моделі та експериментами).

5 Нормалізувати ознаки (наприклад, через стандартизацію або масштабування), щоб числові діапазони були стабільними та зручними для нейронної мережі.

6 Сформувати тензор $X$ фіксованого розміру, узгодженого з CNN: виконати обрізання по часу або доповнення нулями (padding), а також привести форму до потрібної кількості каналів.

7 Виконати інференс нейромережі та отримати вектор імовірностей $p$ (або еквівалентний вихід моделі, що інтерпретується як розподіл по класах).

8 Визначити прогнозований клас як індекс максимального елемента $p$, тобто $\hat{y}$; сформувати відображення результату для користувача (клас, топ-$k$ варіантів, рівень впевненості).

9 Відобразити результат у UI та, за потреби, виконати логування (параметри обробки, версія моделі, час виконання, прогноз).

**Алгоритм попередньої обробки аудіосигналу**

Мета попередньої обробки — зробити сигнал порівнюваним між різними файлами та умовами запису. У практичних наборах даних аудіо відрізняється форматом (моно/стерео), частотою дискретизації, амплітудним рівнем, а також

може містити ділянки тиші чи фоновий шум. Якщо ці відмінності не усунути або хоча б не стабілізувати, модель може “вивчити” випадкові артефакти запису замість корисних ознак класу. Тому передобробка включає приведення каналів, уніфікацію частоти дискретизації, нормалізацію, фреймінг та застосування віконної функції перед спектральним аналізом.

Алгоритм попередньої обробки аудіо:

Вхід: сигнал $x(t)$, частота дискретизації $f_s$.

1 Якщо сигнал записаний у стерео, виконати перехід у моно усередненням каналів:

$$x_{\text{mono}}(t) = \backslash dfrac x_L(t) + x_R(t) 2. \tag{2.2}$$

У випадку, коли сигнал уже моно, цей крок пропускається.

2 Якщо частота дискретизації не відповідає стандарту системи ($f_s \neq f_s^*$), виконати ресемплінг до $f_s^*$, щоб однакові події мали однакову часову/частотну роздільну здатність у всіх прикладах.

3 Нормалізувати амплітуду (наприклад, до діапазону $[-1,1]$), щоб зменшити залежність ознак від гучності запису:

$$x_{\text{norm}}(t) = \backslash dfrac x(t) \max|x(t)| + \varepsilon. \tag{2.3}$$

4 Розбити сигнал на фрейми довжини $L$ із кроком $H$ (перекриття). Це дозволяє аналізувати сигнал локально в часі та формувати ознаки з керованою роздільною здатністю.

5 До кожного фрейму застосувати віконну функцію (наприклад, Геммінга), щоб зменшити спектральні витікання при FFT:

$$x_w[n] = x[n] \cdot w[n] \text{ , } w[n] = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{L-1}\right). \tag{2.4}$$

Вихід: набір підготовлених віконованих фреймів, придатних для спектрального аналізу та формування MFCC/спектрограми.

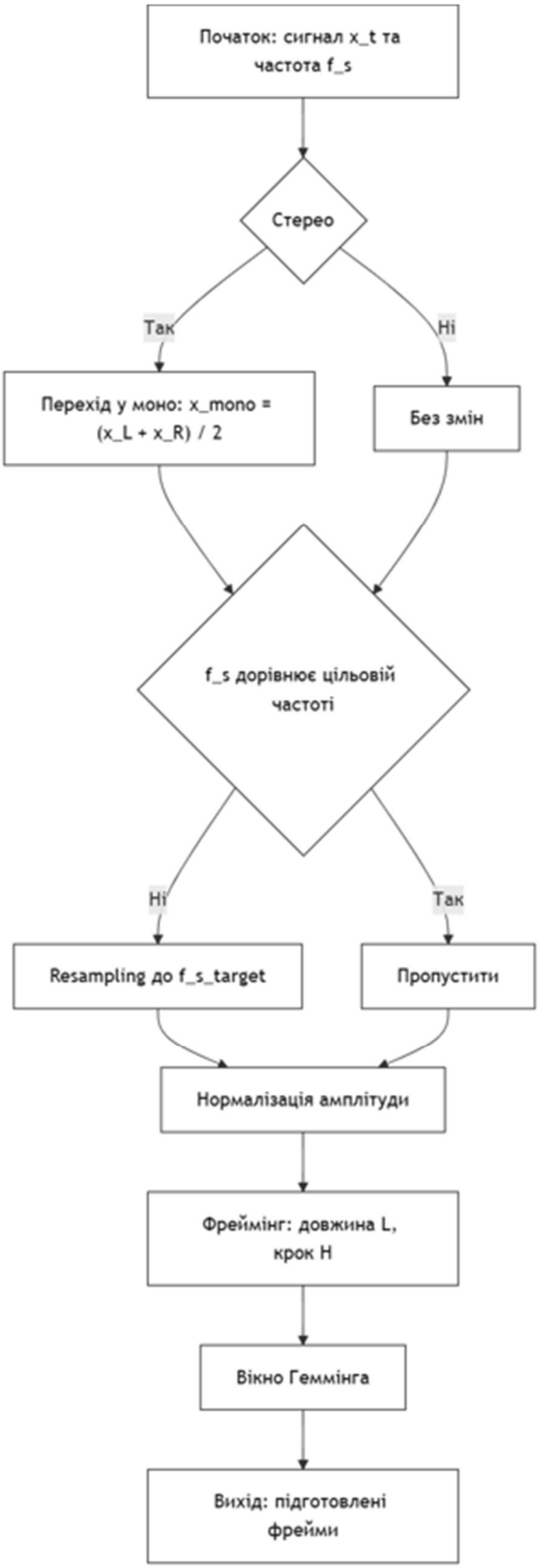


Рисунок 2.2 — Алгоритм попередньої обробки аудіосигналу

**Алгоритм обчислення MFCC для подачі в CNN**

Хоч MFCC детально розглядаються як теоретичний інструмент у попередньому розділі, у контексті проєктування необхідно зафіксувати MFCC як регламентований алгоритм, який система виконує з визначеними параметрами та з чіткою формою виходу. Для CNN важливо не лише “порахувати MFCC”, а й отримати матрицю стабільного формату, яку можна безпосередньо перетворити на тензор входу. Тому, окрім власне обчислення коефіцієнтів, процедура включає агрегацію по часу, нормалізацію та узгодження розмірів (обрізання/паддінг).

Щоб не перевантажувати текст, нижче наведено компактну математичну фіксацію (без надлишкових проміжних формул), але з розширеним поясненням. Для кожного віконованого фрейму $x_w[n]$ виконується FFT та оцінка енергії спектра, після чого застосовується мел-фільтбанк, логарифмування та DCT, що дає MFCC-вектор. Далі MFCC-вектори поєднуються у матрицю ознак.

Алгоритм формування матриці MFCC:

Вхід: набір віконованих фреймів після попередньої обробки.

1 Для кожного фрейму:

– обчислити спектральне представлення (FFT) та отримати спектральну енергію;

– пропустити енергію через мел-фільтбанк і отримати енергії смуг;

– виконати логарифмування енергій (стабілізація масштабу та наближення розподілу);

– застосувати DCT і отримати MFCC-вектор.

2 Об’єднати MFCC-вектори вздовж часової осі у матрицю $C$ (розмірність залежить від кількості фреймів та числа коефіцієнтів):

$$C \in R^{T \times N}, \tag{2.5}$$

де $T$ — кількість фреймів (кроків по часу), $N$ — кількість MFCC-коефіцієнтів (наприклад, $N = 20$).

3 Виконати узгодження розміру матриці для моделі: якщо $T$ перевищує потрібну довжину, матриця обрізається (наприклад, береться центральний фрагмент); якщо $T$ менше — виконується паддінг нулями до фіксованого розміру.

4 Нормалізувати отриману матрицю (локально для файлу або глобально для датасету), щоб забезпечити стабільний діапазон значень на вході CNN.


Рисунок 2.3 — Алгоритм формування MFCC-матриці для CNN

**Алгоритм інференсу CNN та формування рішення**

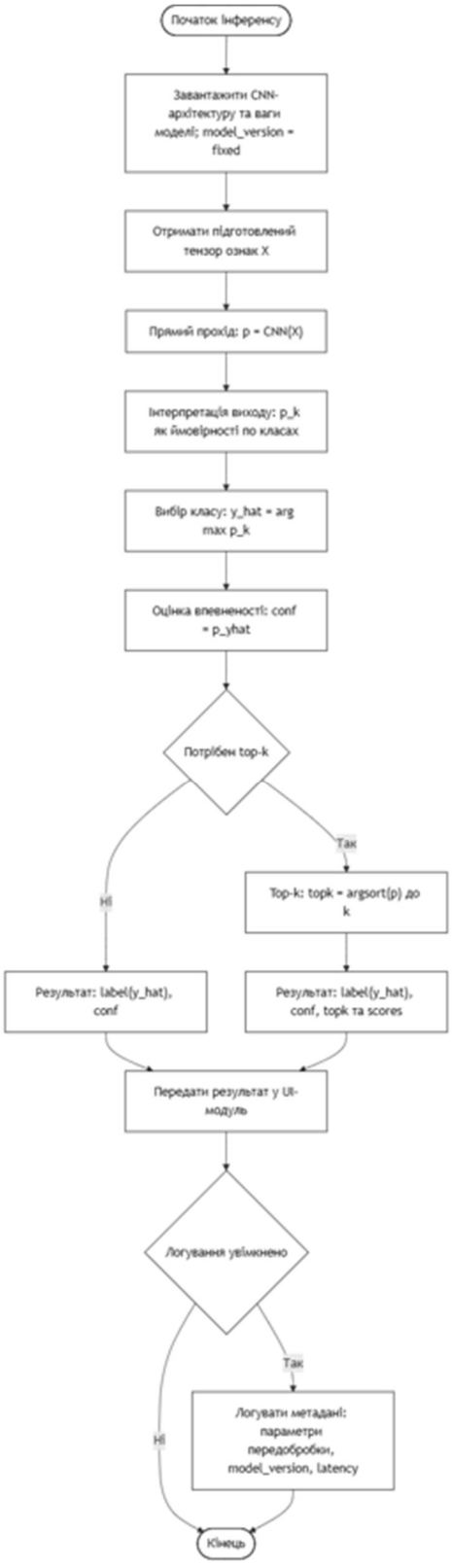


Рисунок 2.4 — Алгоритм інференсу CNN та формування рішення

Етап інференсу виконує перетворення підготовленого тензора ознак у прогноз класу. На практиці це означає, що сформований тензор $X$ передається у модель, яка повертає оцінки по класах у вигляді вектора. Далі ці оцінки інтерпретуються як імовірності, після чого застосовується правило вибору класу. У контексті прикладної системи важливо також передбачити формат видачі результатів: система може повертати один клас (найімовірніший) або кілька найкращих варіантів (top-$k$) разом із відповідними значеннями впевненості. Це підвищує практичну корисність рішення: користувач бачить не лише “ярлик”, а й те, наскільки модель відрізняє домінантний клас від альтернатив.

Для стислої формалізації достатньо зафіксувати правило вибору класу:

$$\hat{y} = \arg\max_{k} p_k. \tag{2.6}$$

Алгоритм інференсу та прийняття рішення:

1 Завантажити архітектуру CNN та ваги моделі (версія моделі фіксується для відтворюваності);

2 Подати підготовлений тензор $X$ у модель та отримати вектор виходу $p$ (ймовірності по класах).

3 Обчислити $\hat{y}$ як клас із найбільшим значенням $p_k$.

4 Сформувати представлення результату: назва класу, значення $p_{\hat{y}}$, а також (за потреби) список top-$k$ альтернатив.

5 Передати результат в модуль UI та (опційно) виконати логування параметрів передобробки й часу виконання, щоб полегшити подальший аналіз помилок.

## 2.3 Проєктування архітектури програмної системи

Проєктування архітектури програмної системи автоматичної класифікації аудіоданих є фундаментальним етапом розробки, оскільки саме на цьому рівні визначається логіка організації обчислювальних процесів, спосіб взаємодії між компонентами та загальна керованість обробки даних. Архітектура системи

повинна забезпечувати повний і безперервний життєвий цикл аудіоінформації — від моменту отримання вхідного файлу до формування остаточного класифікаційного рішення, представленого у зрозумілому для користувача вигляді. Особливу увагу необхідно приділити чіткому розмежуванню відповідальностей між підсистемами, що дозволяє знизити складність реалізації, підвищити надійність програмного коду та створити умови для подальшого масштабування системи.

У межах даної роботи архітектура розглядається не лише як структурна схема взаємодії модулів, а як інженерна модель, яка визначає потоки даних, формати представлення інформації та правила переходу між етапами обробки. З огляду на використання згорткових нейронних мереж, що вимагають значних обчислювальних ресурсів, особливо важливим є відокремлення обчислювального ядра системи від інтерфейсного шару. Такий підхід забезпечує можливість оптимізації, заміни або оновлення алгоритмічних компонентів без необхідності переробки логіки взаємодії з користувачем.

З концептуальної точки зору програмна система автоматичної класифікації аудіоданих розглядається як сукупність взаємопов’язаних програмних компонентів, кожен з яких реалізує окремий етап перетворення інформації. Початково система працює з аудіофайлом у цифровому форматі, який після зчитування перетворюється у дискретний сигнал $x(t)$. Подальші етапи обробки переводять цей сигнал у спектральне представлення, далі — у компактну матрицю ознак, а на фінальному етапі — у тензор, що подається на вхід згорткової нейронної мережі.

Архітектурна побудова ґрунтується на принципі розділення відповідальностей, згідно з яким кожен компонент системи виконує чітко визначену функцію і не виходить за межі своєї логічної ролі. Це дозволяє зменшити зв’язність між модулями та підвищити внутрішню згуртованість кожного з них. Важливим принципом є також розширюваність архітектури, що передбачає можливість підключення нових методів виділення ознак,

альтернативних нейронних моделей або розширення кількості класів без радикальної перебудови програмного ядра. Окремо слід зазначити принцип відтворюваності, який є критично важливим у контексті наукових досліджень: система повинна гарантувати стабільність результатів для однакових вхідних даних та фіксованих параметрів обробки.


Рисунок 2.5 – Компонентна схема програмної системи класифікації аудіоданих

Модульна структура програмної системи формується таким чином, щоб кожен логічний етап обробки аудіоданих був реалізований у вигляді окремого

програмного компонента. Інтерфейс користувача виконує роль зовнішнього шару системи, через який здійснюється взаємодія з користувачем. Він відповідає за вибір аудіофайлу, ініціацію процесу класифікації та відображення результатів, при цьому не містить обчислювальної логіки, пов'язаної з цифровою обробкою сигналів або нейронними мережами.

Після надходження запиту від інтерфейсу керування передається до модуля введення та валідації аудіоданих, який забезпечує зчитування файлу, аналіз його технічних характеристик і перевірку коректності формату. Далі сигнал обробляється модулем попередньої обробки, де виконується приведення даних до уніфікованого вигляду шляхом ресемплінгу до фіксованої частоти дискретизації $f_s^*$, нормалізації амплітуди та сегментації на фрейми з подальшим віконуванням.

Наступний етап реалізується модулем виділення ознак, який формує спектральні характеристики сигналу у вигляді матриці MFCC або mel-спектрограми. Отримані ознаки передаються до адаптера даних, де виконуються операції нормалізації, усічення або доповнення по часовій осі та формування вхідного тензора $X$, сумісного з архітектурою CNN. Модуль інференсу нейронної мережі завантажує попередньо навчену модель, обчислює вектор ймовірностей $p$ і передає його до модуля інтерпретації результатів, який формує прогнозований клас $\hat{y}$ та супровідну інформацію для користувача. Окремий модуль журналювання забезпечує збереження результатів класифікації та параметрів запуску, що дозволяє відновлювати хід експериментів і формувати звіти.

Компонентна схема системи відображає логіку взаємодії між основними програмними модулями та напрямки передачі даних між ними. Вона демонструє, що всі потоки обробки починаються з інтерфейсу користувача і проходять послідовно через модулі введення даних, попередньої обробки, виділення ознак, адаптації даних і нейронного інференсу. Фінальний етап включає інтерпретацію результатів і повернення інформації до користувача, а також запис результатів у

журнал. Така схема наочно підтверджує модульність архітектури та відсутність жорстких залежностей між компонентами, що є важливою перевагою з точки зору підтримки та розвитку системи.

Для формалізації процесу обробки аудіоданих систему доцільно описати як послідовність перетворень із чітко визначеними форматами представлення інформації. Вхідним об’єктом є аудіофайл $f$, який після зчитування переходить у дискретний сигнал $x(t)$. Після спектрального аналізу формується матриця ознак $C \in R^{T \times N}$, що містить часову послідовність спектральних характеристик. На етапі підготовки до нейронної мережі ця матриця перетворюється у тензор $X \in R^{H \times W \times \mathbb{1}}$, який безпосередньо подається на вхід CNN. Результатом роботи мережі є вектор ймовірностей $p \in R^{K}$ та відповідний прогнозований клас $\hat{y}$. Така формалізація потоків даних дозволяє чітко визначити межі відповідальності кожного модуля та спрощує процедури тестування й налагодження.

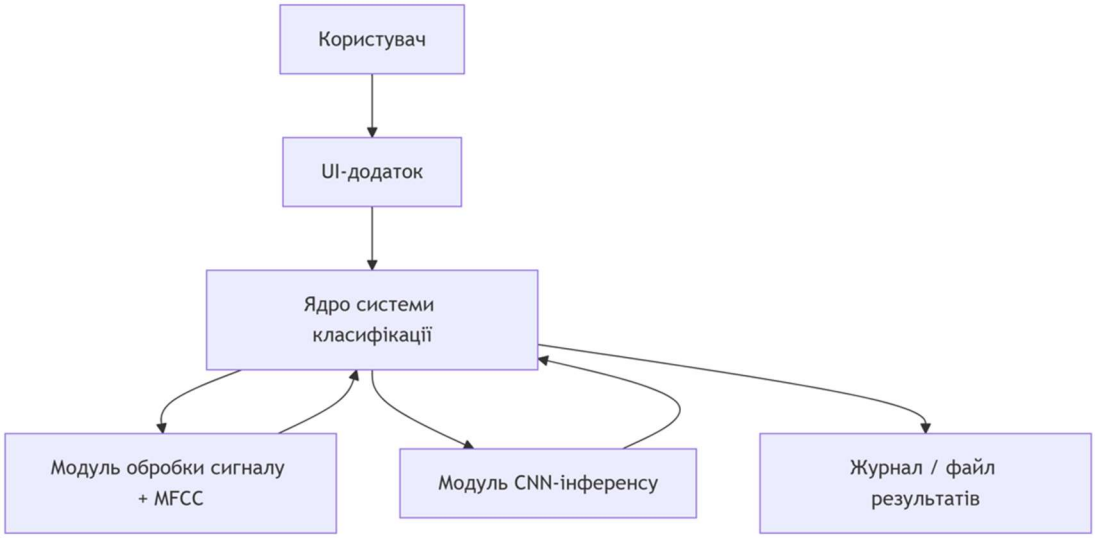


Рисунок 2.6 – Схема розгортання (локальне виконання)

З огляду на вимогу наявності графічного інтерфейсу користувача та автономної роботи системи, у даній роботі обґрунтовується вибір настільної схеми розгортання. Усі компоненти системи функціонують локально на персональному комп’ютері користувача під керуванням сучасної операційної

системи. Реалізація здійснюється мовою програмування Python, що забезпечує доступ до широкого спектра бібліотек для аудіообробки, обчислення MFCC та виконання нейронних мереж. Така технологічна платформа дозволяє досягти прийнятної продуктивності навіть без використання спеціалізованих апаратних прискорювачів і забезпечує простоту встановлення та експлуатації системи.

Запропонована модульна структура забезпечує чітке розмежування функцій між компонентами, підвищує керованість програмного забезпечення та створює умови для подальшого розширення й удосконалення системи. Архітектурні рішення відповідають вимогам інженерної практики та забезпечують стабільність, відтворюваність і надійність результатів класифікації, що є важливим для використання системи як у навчальних, так і в прикладних дослідницьких завданнях.

## 2.4 UML-моделювання та проєктування сценаріїв взаємодії

У процесі проєктування програмної системи автоматичної класифікації аудіоданих важливу роль відіграє UML-моделювання, яке дозволяє формалізувати як зовнішні сценарії взаємодії користувача з системою, так і внутрішню логіку функціонування програмних компонентів. На відміну від суто описового підходу, UML-діаграми надають чітку інженерну інтерпретацію вимог, дозволяючи зафіксувати ролі акторів, відповідальності модулів і порядок виконання операцій. Це особливо актуально для систем, що поєднують алгоритми цифрової обробки сигналів та методи глибокого навчання, оскільки такі системи мають складну багаторівневу структуру.

У межах даної роботи UML-діаграми використовуються не як допоміжні ілюстрації, а як повноцінний засіб проєктування. Вони забезпечують однозначність трактування сценаріїв використання, спрощують перевірку коректності архітектурних рішень і створюють основу для подальшої реалізації програмного коду. Для опису системи визнано достатнім використання трьох

типів UML-діаграм: діаграми варіантів використання (Use Case), діаграми послідовностей (Sequence) та діаграми активностей (Activity), які разом охоплюють функціональний, часовий та логічний аспекти роботи системи.


Рисунок 2.7 – Use Case-діаграма програмної системи класифікації аудіоданих

Діаграма варіантів використання відображає функціональні можливості системи з точки зору користувача та визначає перелік дій, які можуть бути ініційовані під час взаємодії з програмним продуктом. Основним актором у системі є Користувач, який взаємодіє з програмою через графічний інтерфейс та ініціює процеси завантаження аудіофайлу, класифікації та перегляду результатів. Разом з тим, з інженерної точки зору, система передбачає наявність

внутрішніх функцій, які не ініціюються безпосередньо користувачем, але є критично важливими для виконання основного сценарію.

Use Case-діаграма дозволяє наочно відобразити, що запуск класифікації є складеним варіантом використання, який логічно включає виділення ознак MFCC та виконання інференсу згорткової нейронної мережі. Користувач не взаємодіє з цими етапами безпосередньо, проте вони є обов'язковими складовими процесу. Крім базового сценарію, система передбачає можливість перегляду розширених результатів, зокрема списку найбільш імовірних класів із відповідними значеннями ймовірностей, що підвищує інформативність і прозорість роботи класифікатора.

Рисунок 2.7 ілюструє Use Case-діаграму програмної системи класифікації аудіоданих, подану у вигляді коду для подальшого відтворення у зовнішніх UML-редакторах.

Для забезпечення однозначності реалізації та перевірки коректності роботи системи основний сценарій використання фіксується у вигляді формалізованої послідовності дій. Центральним сценарієм є процес класифікації аудіофайлу, який починається з ініціації користувачем і завершується відображенням результату. Передумовами виконання сценарію є наявність вибраного аудіофайлу, готовність програмної системи до роботи та доступність попередньо навчених ваг моделі CNN.

У межах цього сценарію система зчитує аудіосигнал, визначає його параметри та приводить дані до уніфікованого формату, що включає перетворення у моно та ресемплінг до фіксованої частоти дискретизації $f_s^*$. Після цього виконується сегментація сигналу на фрейми, застосування віконної функції та обчислення матриці MFCC. Отримані ознаки трансформуються у тензор $X$, який подається на вхід згорткової нейронної мережі. Результатом інференсу є вектор ймовірностей $p$, на основі якого визначається прогнозований клас $\hat{y}$. За необхідності результат зберігається у журналі для подальшого аналізу або формування звітів.

Діаграма послідовностей використовується для відображення часової взаємодії між об’єктами системи під час виконання основного сценарію. Вона дозволяє простежити, як ініціатива користувача через інтерфейс послідовно передається до модулів введення аудіо, попередньої обробки, виділення ознак, адаптації даних, нейронного інференсу та обробки результатів. Така діаграма є особливо корисною для реалізації, оскільки вона прямо відповідає структурі викликів методів у програмному коді.

На рисунку 2.8 наведено Sequence-діаграму сценарію «Класифікувати аудіофайл».

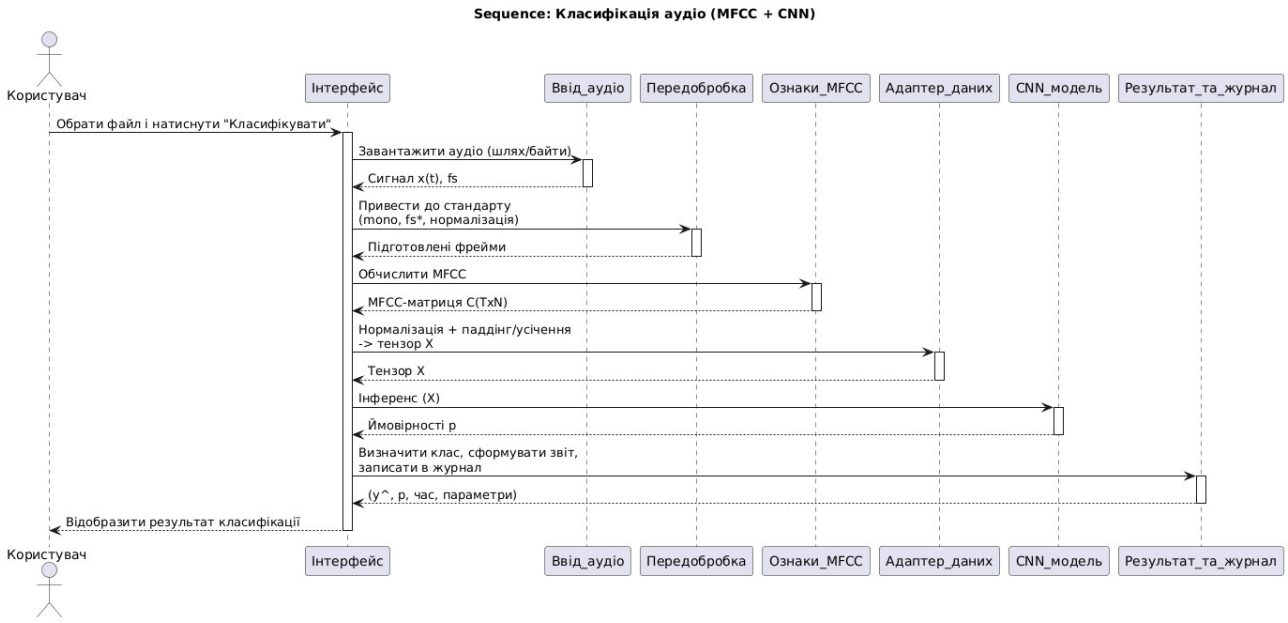


Рисунок 2.8 – Sequence-діаграму сценарію «Класифікувати аудіофайл».

Діаграма активностей дозволяє описати логіку виконання процесу класифікації з урахуванням умовних переходів і можливих альтернатив. Вона відображає перевірки підтримуваного формату аудіо, допустимої тривалості сигналу, вибір подальшого шляху обробки у разі необхідності усічення або

сегментації. Така діаграма є корисною для аналізу граничних випадків і помилкових ситуацій, які можуть виникати під час експлуатації системи.

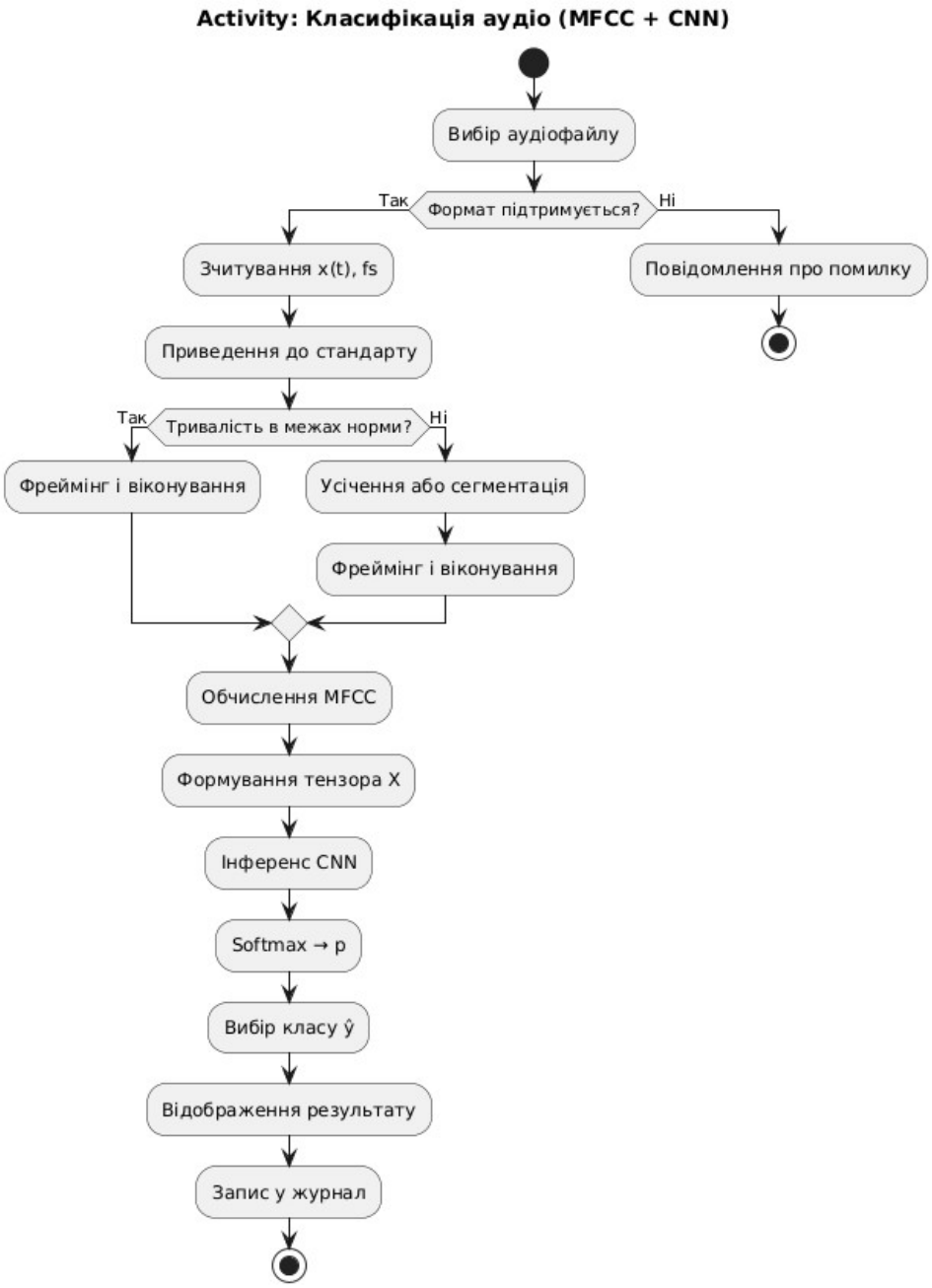


Рисунок 2.9 – Sequence-діаграму сценарію «Класифікувати аудіофайл

Рисунок 2.9 демонструє Activity-діаграму процесу класифікації аудіоданих.

Діаграма станів корисна, якщо в системі є інтерфейс із керованими режимами: «немає файлу», «файл завантажено», «йде класифікація», «є результат», «помилка», «скидання». Вона формалізує допустимі переходи та спрощує реалізацію коректної поведінки кнопок/полів.

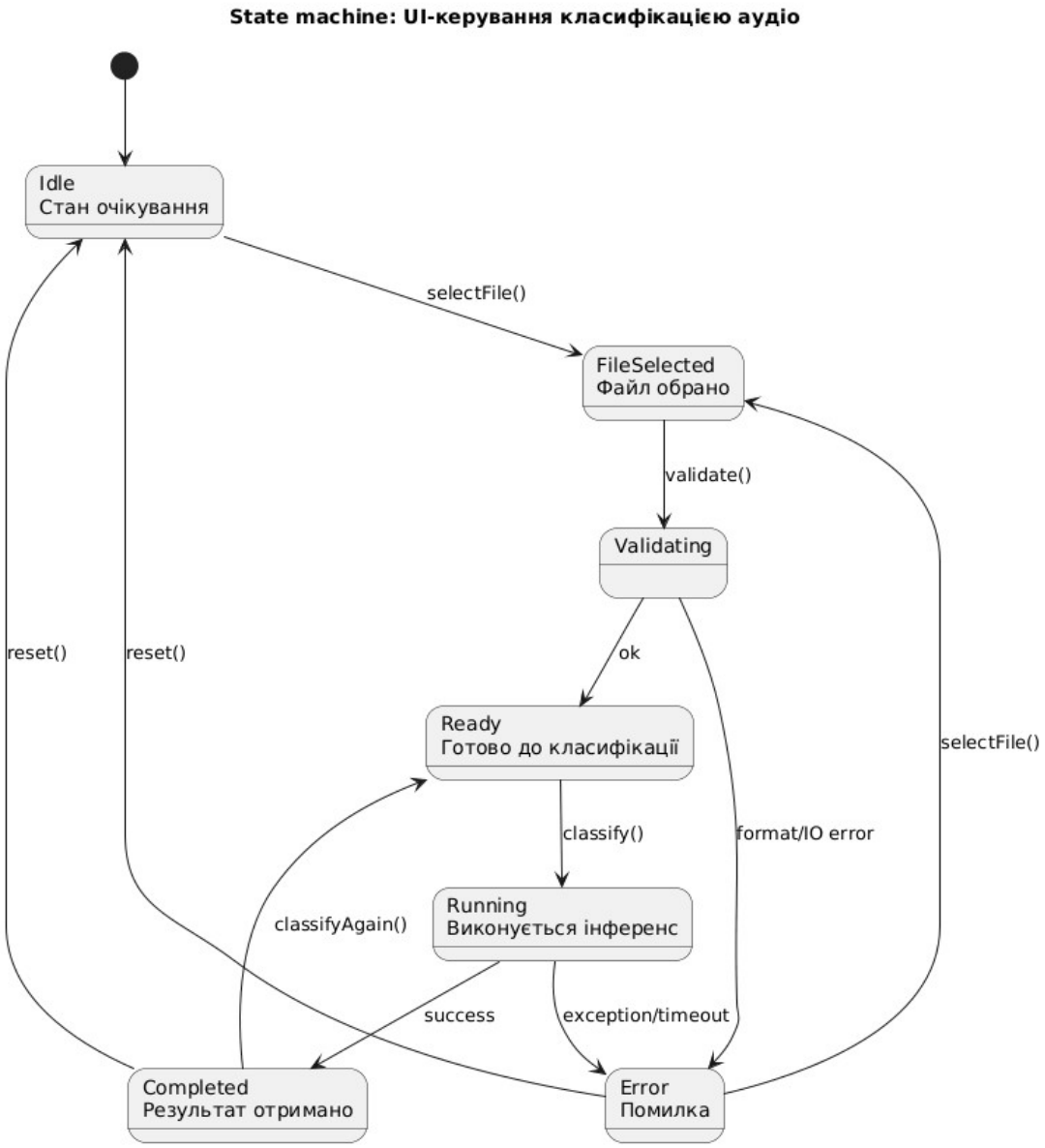


Рисунок 2.10 – Діаграма станів

Описово доцільно зазначити, що стан Running має бути атомарним з точки зору UI: у цей момент користувач не повинен повторно запускати інференс або

змінювати файл, інакше виникають гонки та неконсистентні результати. Такі вимоги безпосередньо випливають із діаграми станів і можуть бути реалізовані блокуванням елементів UI.

### 2.5 Модель програмних класів та відповідальності компонентів

Для реалізації модульної архітектури доцільно застосувати підхід, за яким кожний етап конвеєра відповідає окремому класу або невеликій групі класів із чітко визначеними межами відповідальності. Таке розділення дозволяє формалізувати точки входу й виходу для кожного етапу, уніфікувати контракти взаємодії, а також спростити перевірку коректності обробки шляхом незалежного тестування модулів. На практиці це означає, що помилка у виділенні ознак або інференсі моделі не «розповзається» на інтерфейс користувача чи механізм журналювання, а локалізується в одному компоненті та виправляється без ризику порушення іншої логіки. Додатковою перевагою є повторне використання модулів: наприклад, препроцесор і екстрактор ознак можуть бути застосовані не лише для поточної CNN, а й для порівняльних експериментів із іншими моделями або для розширення системи на нові класи.

Пропонується виділити такі основні класи, де кожен клас реалізує «інженерно завершений» фрагмент конвеєра, має передбачувані вхідні параметри та повертає стандартизований результат:

1 AudioLoader — відповідає за завантаження аудіофайлу з файлової системи або з іншого джерела, зчитування дискретного сигналу та базових метаданих, а також попереднє узгодження формату з внутрішнім поданням системи. На рівні реалізації цей клас повинен забезпечити коректну роботу з різними контейнерами й кодеками (WAV/PCM, MP3 тощо), виконати декодування у масив відліків та сформувати структурований об’єкт AudioData, у якому зберігаються значення samples, частота дискретизації, кількість каналів та

тривалість. Таким чином, AudioLoader виступає «входом» системи та гарантує, що всі наступні етапи отримують уніфікований сигнал.

2 AudioValidator — реалізує перевірку коректності вхідних даних і накладає обмеження, визначені вимогами (підтримувані формати, мінімальна/максимальна тривалість, допустимі діапазони частоти дискретизації, перевірка на порожній або пошкоджений файл). Важливо, що валідація не повинна змішуватись із препроцесингом: її мета — прийняти або відхилити вхід та надати зрозуміле повідомлення про причину відмови. З інженерної точки зору це знижує кількість «німих» збоїв у глибині конвеєра та забезпечує передбачувану поведінку UI, зокрема у випадках, коли користувач намагається завантажити надто довгий запис або файл з некоректними заголовками.

3 Preprocessor — виконує підготовку сигналу до виділення ознак і вирішує завдання стабілізації параметрів, щоб мінімізувати вплив різнорідності даних. Типовий набір операцій включає перетворення в моно, ресемплінг до цільової частоти $f_S^*$, нормалізацію амплітуди, фреймінг із заданими параметрами $L$ та $H$, а також віконування. Результатом роботи препроцесора є набір фреймів Frames, що вже придатні для спектральних перетворень і формування MFCC/спектрограм, причому всі параметри фреймінгу є контрольованими та однаковими для тренування і для застосування моделі.

4 FeatureExtractor — формує ознаки (MFCC або mel-спектрограму) та забезпечує узгодженість параметрів перетворення. У контексті системи важливо розглядати цей клас як «точку стандартизації ознак», де визначаються значення $N$ (кількість MFCC), $M$ (кількість mel-фільтрів), розмір FFT, схема логарифмування енергій, а також правила обробки крайових випадків (наприклад, додавання ε перед логарифмуванням). Оскільки система може мати два режими (MFCC або mel-спектрограма), FeatureExtractor повинен формувати однакові за контрактом об’єкти FeatureMatrix і надавати можливість відтворювано повторювати обчислення за фіксованою конфігурацією.

5 TensorAdapter — реалізує перехід від ознак у вигляді матриці до тензора, який очікує CNN. Тут зосереджуються операції нормалізації (наприклад, стандартизація), приведення розмірів до фіксованого $T^*$ через паддінг або усічення, а також формування канального виміру. Саме на цьому етапі важливо забезпечити строгий інваріант: незалежно від того, якою була тривалість аудіо або кількість фреймів $T$, на виході формується тензор $X \in R^{T^* \times N \times 1}$, що гарантує сумісність із моделлю та усуває клас помилок, пов'язаних із «плаваючими» розмірами.

6 ModelInference — інкапсулює завантаження CNN-моделі, перевірку доступності ваг, підготовку середовища інференсу та виконання прогнозування для вхідного тензора $X$. Цей модуль повинен повертати вектор ймовірностей класів $p \in R^K$, де $K$ — кількість класів, і забезпечувати однозначну відповідність між індексами вектора та мітками класів через файл labels_path або інший механізм. Важливо, щоб ModelInference не містив UI-логіки та не приймав рішень щодо порогів: його роль — обчислити числовий результат інференсу в уніфікованому форматі.

7 DecisionEngine — реалізує правило прийняття рішення поверх вектора ймовірностей, включно з вибором фінального класу та формуванням «топ-$N$» результатів. На рівні базового рішення достатнім є правило $\hat{y} = \arg\max_k p_k$, але у прикладних системах часто потрібна додаткова логіка: наприклад, оцінка «впевненості» як $\max(p)$, застосування порогу прийняття рішення або повернення попередження у випадку низької впевненості. Відокремлення цього класу дозволяє надалі модифікувати політику рішень без втручання в блоки DSP або CNN.

8 ResultPresenter — готує дані до відображення в інтерфейсі користувача, перетворюючи «сирі» числові результати в читабельний висновок. Його відповідальність включає формування текстового повідомлення про передбачений клас, підготовку таблиці ймовірностей або списку топ-3, а також узгодження формату відображення з вимогами документації (терміни

українською, одиниці вимірювання, правила підпису тощо). Такий модуль важливий, оскільки UI часто змінюється, і бажано, щоб правила представлення результату не були «розкидані» по кнопках/обробниках подій.

9 Logger — забезпечує журналювання запусків класифікації та зберігає контекст, необхідний для відтворюваності експериментів. У записі логів доцільно фіксувати дату/час, ім’я файлу, параметри препроцесингу, тип ознак, версію моделі, результат $\hat{y}$ та впевненість. Окремим аспектом є контроль «відбитка конфігурації» через params_hash, який дозволяє однозначно ідентифікувати умови запуску навіть тоді, коли параметрів багато і вони можуть змінюватись у ході розробки.

10 Config — централізовано зберігає всі параметри обробки та моделі й виступає «єдиним джерелом істини» для конвеєра. Такий клас мінімізує ризик розсинхронізації, коли різні модулі використовують різні значення $f_s^*$, $L$, $H$ або $N$, а також спрощує експерименти, оскільки зміна конфігурації виконується в одному місці. З погляду програмної інженерії Config робить систему більш керованою та придатною до масштабування.

У підсумку запропонований розподіл відповідає типовим практикам побудови ML/Audio систем, де ядро перетворень сигналу та інференс моделі чітко відокремлені від шару представлення.

Об’єкт AudioData виступає контейнером сигналу та базових параметрів, що дозволяє уникати передачі «розсипом» кількох змінних між модулями та підтримувати єдиний формат даних на всьому шляху обробки. Абстракції Frames і FeatureMatrix у проєктному описі наведені як окремі сутності, хоча в реалізації це можуть бути NumPy-масиви або тензори, проте винесення їх на діаграму фіксує факт наявності проміжних представлень, які можна перевіряти й валідовувати. Клас Config використовується більшістю модулів, оскільки містить стандартизовані параметри, що визначають як DSP-частину (частота дискретизації, фреймінг), так і ML-частину (шлях до моделі, режим ознак). Клас Logger взаємодіє зі структурою LogRecord, завдяки чому результати класифікації

отримують «обрамлення» контекстом експерименту, а отже стають придатними для подальшої аналітики, порівняння запусків і підготовки звітності. Важливо, що DecisionEngine і ResultPresenter не залежать від внутрішньої реалізації CNN, а працюють із вектором ймовірностей як з узагальненим контрактом, що підсилює розширюваність системи при заміні моделі.

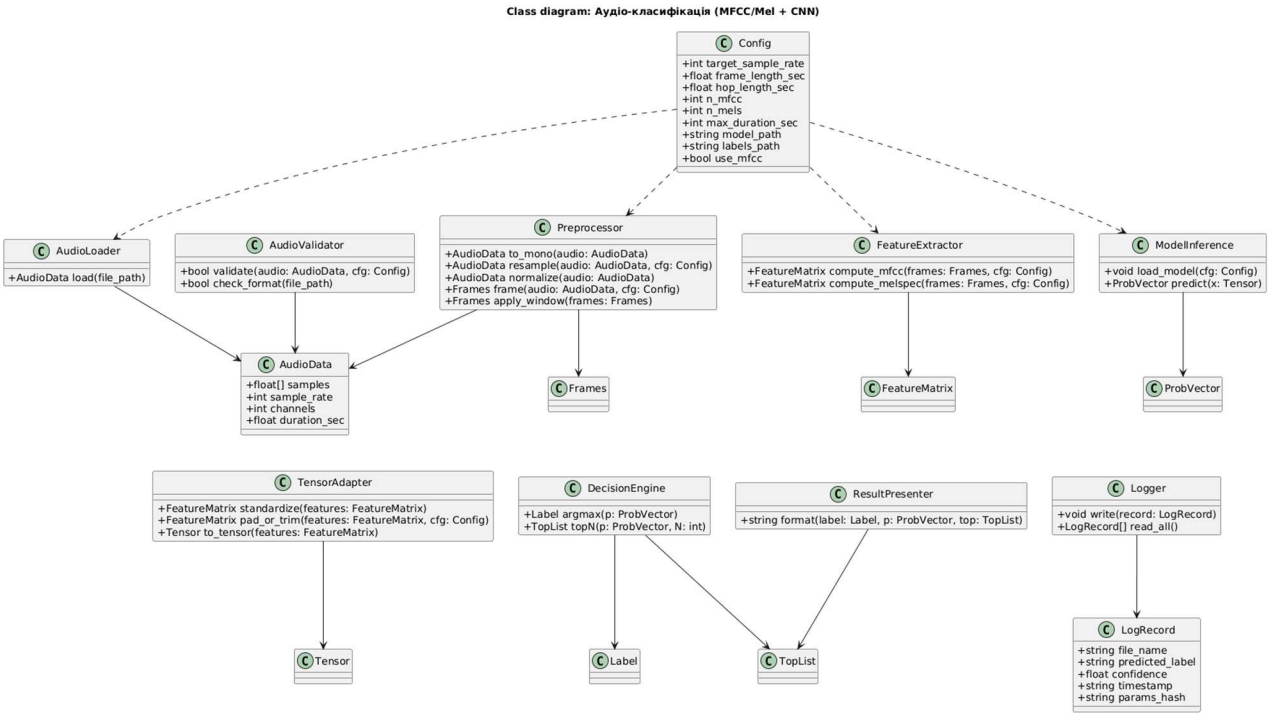


Рисунок 2.11 – Діаграма класів програмної системи класифікації аудіоданих

Для практичної цінності системи важливо не лише вивести результат на екран, а й забезпечити його фіксацію у зручному форматі, придатному для накопичення статистики, повторних експериментів та перевірки стабільності роботи. Журналювання стає особливо значущим у задачах аудіокласифікації, де на вихід може впливати багато факторів: довжина уривка, режим нарізки на фрейми, параметри мел-фільтробанку, вибір кількості MFCC, а також версія моделі й навіть бібліотечні залежності. Тому формат збереження має підтримувати як «ядро» результату (передбачений клас і ймовірності), так і супровідні метадані, необхідні для відтворення умов запуску.

Формат збереження результатів класифікації доцільно реалізувати у вигляді текстового формату, який легко аналізується програмно і водночас читається людиною. На практиці часто використовують CSV, оскільки він простий і сумісний зі spreadsheet-інструментами, однак для задач, де потрібно зберігати вкладені структури (наприклад, топ-3 класи з ймовірностями), більш доречним є JSON. Вибір JSON обґрунтований тим, що в подальшому до запису можна додати нові поля (наприклад, час обчислення MFCC або затримку інференсу) без необхідності змінювати схему таблиць чи ламати сумісність із попередніми логами.

Приклад структури запису журналу (LogRecord) може мати такий вигляд:

```
{
  "timestamp": "2025-12-12 14:37:10",
  "file_name": "audio_001.wav",
  "sample_rate": 16000,
  "duration_sec": 3.2,
  "predicted_label": "SIREN",
  "confidence": 0.92,
  "top3": [
    {"label": "SIREN", "p": 0.92},
    {"label": "CAR_HORN", "p": 0.05},
    {"label": "ENGINE", "p": 0.03}
  ],
  "features": "MFCC",
  "model_version": "cnn_v1",
  "params_hash": "b6a1d2..."
}
```

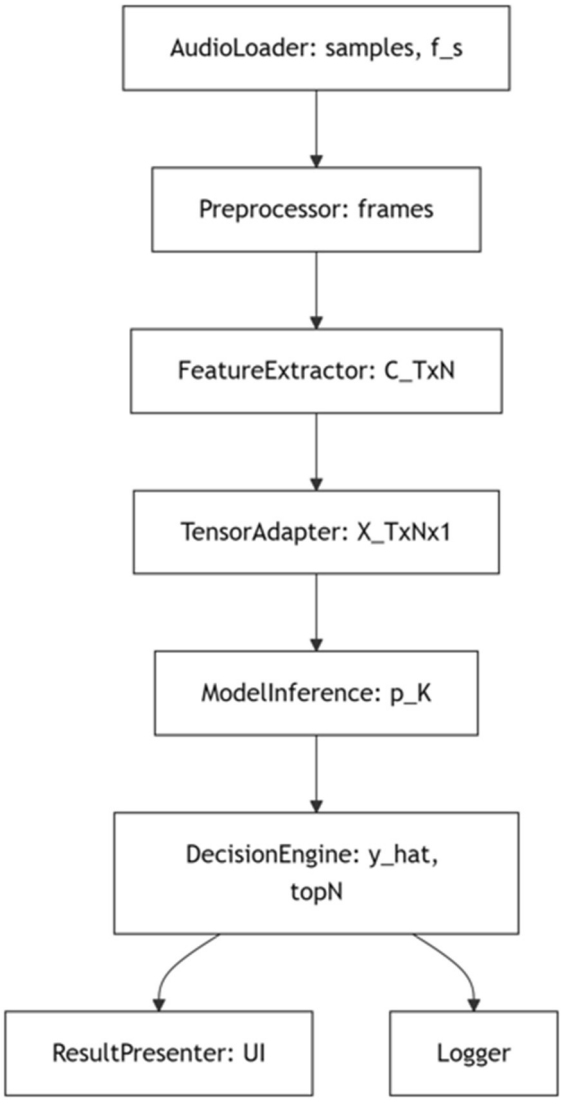


Рисунок 2.12 – Формати даних між модулями системи

У такій структурі журнал містить не лише результат, а й контекст запуску, що критично для інженерної відтворюваності: частота дискретизації та тривалість дозволяють перевірити, що препроцесинг відпрацював у межах обмежень, тип ознак фіксує режим обчислення, версія моделі дозволяє порівнювати різні навчання, а params_hash пов’язує запис із конкретною конфігурацією. Наявність поля top3 робить журнал придатним для якісного аналізу помилок: навіть якщо передбачення неправильне, можна оцінити, чи був

правильний клас «близько» за ймовірністю, що часто вказує на проблему дискретизації, шум або неоднозначність зразка.

Щоб забезпечити контроль коректності реалізації, доцільно визначити, у якому вигляді дані передаються між модулями. Такий опис корисний не лише для пояснювальної записки, а й для практики тестування: можна створювати «контрольні точки» і перевіряти проміжні результати на відповідність формату та діапазону значень. Наприклад, після нормалізації сигнал має бути у діапазоні близькому до $[-1,1]$, після екстракції MFCC матриця повинна мати очікувані розміри $T \times N$, а після адаптації — фіксований $T^* \times N \times 1$.

У цій схемі кожен перехід можна трактувати як контракт. AudioLoader гарантує, що AudioData містить коректний масив відліків і метадані; Preprocessor перетворює сигнал на множину фреймів, узгоджених за довжиною; FeatureExtractor формує матрицю ознак, де семантика осей є фіксованою (час × коефіцієнти); TensorAdapter приводить представлення до фіксованого формату тензора для CNN; ModelInference повертає вектор ймовірностей, який вже є достатнім для роботи DecisionEngine; останній формує кінцевий клас $\hat{y}$ і допоміжні результати, які або відображаються користувачу, або фіксуються в журналі.

У другому розділі виконано проєктування програмної системи класифікації аудіоданих та формалізовано повний конвеєр обробки від аудіофайлу до кінцевого рішення моделі. На підставі постановки задачі визначено й структуровано функціональні вимоги до ключових підсистем: введення та валідації аудіофайлів, попередньої обробки сигналу, виділення акустичних ознак, модулю нейромережевої класифікації та інтерфейсу користувача. Показано, що для забезпечення відтворюваності й стабільності результатів критично важливими є уніфікація параметрів аудіо (моно, ресемплінг, нормалізація), фреймінг із перекриттям та застосування віконних функцій перед спектральними перетвореннями.

Розроблено алгоритмічне забезпечення системи у вигляді формалізованих процедур: загального конвеєра класифікації, алгоритму попередньої обробки, формування MFCC-матриці для подачі в CNN та алгоритму інференсу з правилом вибору класу на основі максимуму ймовірності. Зафіксовано вимоги до узгодження розмірностей (обрізання/доповнення по часу, додавання канального виміру), нормалізації ознак і формування тензора фіксованого формату, що гарантує сумісність із архітектурою згорткової мережі та знижує ризик помилок, пов’язаних із “плаваючими” розмірами вхідних даних.

Запропоновано модульну архітектуру, побудовану на принципі розділення відповідальностей між компонентами: завантаження та валідація аудіо, препроцесинг, екстракція ознак, адаптація до тензорного формату, інференс CNN, прийняття рішення, представлення результатів і журналювання. Визначено потоки даних та їх формати на межах модулів, що створює підґрунтя для модульного тестування й подальшого розширення системи (заміна типу ознак, зміна кількості класів, підключення альтернативної моделі). UML-моделюванням формалізовано сценарії використання та послідовність взаємодії компонентів у базовому сценарії «Класифікувати аудіофайл», а також описано логіку обробки альтернативних і помилкових ситуацій.

Окремо обґрунтовано підхід до фіксації результатів класифікації через журналювання у структурованому форматі (наприклад, JSON), що забезпечує інженерну відтворюваність експериментів завдяки збереженню контексту запуску: параметрів аудіо, режиму ознак, версії моделі та показників упевненості. Таким чином, у розділі сформовано завершену проєктну основу для практичної реалізації системи: визначено вимоги, алгоритми, архітектуру, моделі взаємодії та формати даних, які будуть безпосередньо використані у наступному розділі під час реалізації й експериментального оцінювання ефективності.

# 3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ ТА РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

У третьому розділі описується практична реалізація програмної системи автоматичної класифікації аудіоданих на основі згорткової нейромережі, а також результати експериментального дослідження її ефективності. Реалізація здійснюється відповідно до архітектури та алгоритмічних рішень, спроєктованих у розділі 2, із дотриманням модульного підходу. Ключова ідея полягає в тому, що обчислювально складні процедури (попередня обробка сигналу, формування ознак, інференс моделі) відокремлюються від інтерфейсного шару, тоді як взаємодія між компонентами будується через чітко визначені структури даних. Така організація не лише спрощує програмну реалізацію, але й робить систему керованою з позицій тестування та подальшого супроводу: кожний модуль можна перевіряти автономно, а помилки локалізувати на рівні конкретного етапу конвеєра.

Для забезпечення відтворюваності експериментів у реалізації передбачено конфігураційні параметри, що фіксують основні режими роботи системи, зокрема частоту дискретизації, тривалість фрейму, крок фреймування, кількість MFCC-коефіцієнтів та максимально допустиму тривалість запису. Додатково реалізується журналювання результатів класифікації, яке є важливим інженерним механізмом: воно дозволяє пов’язати отриманий клас із конкретним файлом, параметрами запуску та значенням впевненості моделі, а отже — надалі виконувати аналіз типових помилок, повторювати експерименти та формувати підсумкові звіти на основі накопичених даних. Експериментальна частина розділу передбачає оцінювання якості класифікації за стандартними метриками, а також якісну інтерпретацію помилок, які можуть виникати через шум, неоднорідність записів або недостатню вираженість ознак у коротких фрагментах.

## 3.1 Обрані технології та засоби реалізації

Для реалізації програмної системи автоматичної класифікації аудіоданих обрано сучасний стек програмних і алгоритмічних засобів, що забезпечують високу точність обробки сигналів, відтворюваність експериментів і практичну придатність розробленого рішення.

Як основну мову програмування використано Python, що зумовлено його широким застосуванням у галузі машинного навчання, наявністю розвиненої екосистеми бібліотек для цифрової обробки сигналів і глибокого навчання, а також зручністю прототипування та інтеграції компонентів.

Для попередньої обробки аудіосигналів і виділення інформативних ознак застосовано метод мел-частотних кепстральних коефіцієнтів (MFCC), який є де-факто стандартом у задачах аудіоаналізу та мовної обробки. MFCC забезпечують компактне та стійке до шумів представлення спектральних властивостей сигналу, що добре узгоджується з подальшим використанням нейромережевих моделей.

Як основний класифікатор використано згорткову нейронну мережу (CNN), реалізовану засобами фреймворку TensorFlow / Keras. Такий вибір обґрунтований здатністю CNN ефективно виявляти локальні частотно-часові патерни у двовимірних представленнях аудіоданих (MFCC), а також підтримкою сучасних механізмів регуляризації, оптимізації та збереження моделей.

Для реалізації графічного інтерфейсу користувача застосовано бібліотеку Tkinter, що входить до стандартного набору Python. Це забезпечує простоту розгортання, незалежність від сторонніх UI-фреймворків і можливість створення зрозумілого настільного інтерфейсу для демонстрації результатів класифікації.

Додатково використано бібліотеки NumPy та SciPy для чисельних обчислень і цифрової обробки сигналів, а також scikit-learn для обчислення метрик якості та формування звітів. Така комбінація технологій забезпечує

цілісність програмної системи, узгодженість теоретичних і практичних рішень та можливість подальшого розширення функціональності.

### 3.2 Програмна реалізація модулів обробки аудіоданих

Програмний комплекс реалізовано як сукупність взаємодіючих модулів, які послідовно виконують перетворення даних від моменту отримання аудіофайлу до формування кінцевого рішення. На практичному рівні система працює як єдиний конвеєр, у якому кожний крок має чітко визначений вхід і вихід: спочатку файл зчитується та валідується, далі сигнал приводиться до стандартизованих параметрів, після чого обчислюються ознаки (у даному випадку MFCC або споріднене представлення), отримана матриця ознак нормалізується і перетворюється у тензор фіксованого розміру, придатний для подачі в згорткову нейромережу, а вже на завершальному етапі виконується інференс CNN, формується вектор ймовірностей класів та приймається рішення щодо прогнозованої мітки. Такий порядок не є формальністю: він відображає технологічну необхідність забезпечити однакову обробку як тренувальних даних (на етапі побудови моделі), так і реальних користувацьких файлів (на етапі експлуатації), тобто гарантувати узгодженість розподілів ознак і коректність інференсу.

Модуль введення реалізує початковий етап роботи системи, який полягає у взаємодії з аудіофайлом та формуванні первинного представлення сигналу у вигляді масиву відліків. Саме на цьому етапі доцільно отримати та зафіксувати параметри, без яких подальша обробка буде некоректною або неоднозначною. До таких параметрів належать частота дискретизації $f_s$, кількість каналів та тривалість запису, адже вони визначають, чи може файл бути оброблений поточними налаштуваннями системи і чи не порушуються обмеження на розмір вхідних даних. Валідація виконується якомога раніше, оскільки це дозволяє уникнути зайвих обчислень у випадках, коли файл має непідтримуваний формат

або його тривалість виходить за допустимі межі. У практичній експлуатації така перевірка суттєво підвищує надійність: система не “падає” на середині обчислення MFCC або інференсу, а повертає зрозуміле повідомлення про причину відмови ще до початку ресурсоємних операцій.

З точки зору структури даних, для подальшої узгодженої роботи модулів зручно використовувати контейнерний клас AudioData, який містить масив відліків, частоту дискретизації, кількість каналів і тривалість. Це зменшує кількість “розкиданих” параметрів у сигнатурах функцій і робить передачу даних між модулями більш контрольованою. Валідація у наведеному нижче прикладі зводиться до перевірки розширення файлу та базової перевірки тривалості, однак у повній реалізації вона може бути розширена контролем мінімальної тривалості, перевіркою на порожній сигнал, а також контролем допустимого діапазону $f_s$.

Фрагмент коду, що ілюструє структуру даних та базову валідацію, наведено нижче:

```
from dataclasses import dataclass
import numpy as np
@dataclass
class AudioData:
    samples: np.ndarray
    sample_rate: int
    channels: int
    duration_sec: float
class AudioValidator:
    def __init__(self, allowed_ext=("wav", "mp3",
"flac")):
        self.allowed_ext = allowed_ext

    def check_format(self, file_path: str) -> bool:
        ext = file_path.lower().split(".")[-1]
```

```
            return ext in self.allowed_ext
        def validate_basic(self, audio: AudioData,
max_duration_sec: int) -> bool:
            return audio.duration_sec > 0 and
audio.duration_sec <= max_duration_sec
```

У випадку, якщо аудіо не відповідає вимогам (наприклад, має надмірну тривалість), система не повинна “ігнорувати” проблему, оскільки довгі записи можуть призводити до неконтрольованого зростання числа фреймів $T$ та збільшення часу обчислення ознак. Тому доцільно передбачати одну з двох політик: або усікати сигнал до допустимої тривалості, або виконувати сегментацію та класифікацію по фрагментах із подальшою агрегацією (наприклад, середнім значенням ймовірностей або правилом більшості). Конкретний вибір політики визначається конфігурацією та вимогами сценарію використання: якщо потрібна швидкість і простота, використовують усічення; якщо важливіша стабільність рішення для довгих записів, застосовують сегментацію.

Попередня обробка аудіосигналу в системі виконує роль “вирівнювача” даних, який зменшує вплив відмінностей умов запису і забезпечує узгодженість подальших ознак. У практичних даних навіть записи одного класу можуть суттєво відрізнятися за амплітудою, співвідношенням сигнал/шум, форматом (моно/стерео) та частотою дискретизації. Якщо ці фактори не стабілізувати, то виділені ознаки можуть відображати не сутність класу, а технічні відмінності записів, що призводить до погіршення узагальнювальної здатності моделі. Саме тому на етапі попередньої обробки система приводить сигнал до стандартного виду: за потреби виконує перетворення у моно, здійснює ресемплінг до цільової частоти $f_S^*$, нормалізує амплітуду, а також готує сигнал до спектрального аналізу через фреймінг.

В фрагменті коду нижче продемонстровано базові операції, які є типовими для модуля попередньої обробки. Перехід у моно реалізовано через усереднення каналів, що є стандартним підходом для зменшення розмірності без втрати загальної енергетики сигналу. Нормалізація виконується за описаним правилом, а операція trim_or_pad дозволяє привести сигнал до фіксованої довжини, що є важливим для стабільності вхідних даних, особливо якщо система працює з короткими фрагментами або потребує уніфікованої тривалості вхідного сегмента.

```
    import numpy as np
    class Preprocessor:
        def __init__(self, target_sr: int = 16000, eps: 
float = 1e-9):
            self.target_sr = target_sr
            self.eps = eps
        def to_mono(self, samples: np.ndarray) -> 
np.ndarray:
            if samples.ndim == 1:
                return samples
            return samples.mean(axis=1)
        def normalize(self, samples: np.ndarray) -> 
np.ndarray:
            m = np.max(np.abs(samples))
            return samples / (m + self.eps)
        def trim_or_pad(self, samples: np.ndarray, 
target_len: int) -> np.ndarray:
            if len(samples) >= target_len:
                return samples[:target_len]
            pad = target_len - len(samples)
            return np.pad(samples, (0, pad), 
mode="constant")
```

Практичне значення попередньої обробки проявляється і на етапі тестування системи. Якщо у контрольній вибірці присутні записи з різною частотою дискретизації або різною гучністю, то відсутність ресемплінгу та нормалізації може призводити до того, що модель демонструватиме нестабільні результати навіть на однакових за змістом аудіофрагментах. Натомість стандартизація параметрів зменшує розкид ознак і робить поведінку класифікатора більш передбачуваною. Саме тому в експериментальній частині доцільно фіксувати, які параметри попередньої обробки використовувалися, та, за можливості, порівнювати результати для різних налаштувань (наприклад, різні $f_s^*$ або різні довжини сегмента), щоб обґрунтувати вибір конфігурації як компроміс між якістю та швидкодією.

### 3.3 Реалізація модуля виділення ознак MFCC

У цьому підрозділі наведено програмну реалізацію модуля виділення мел-частотних кепстральних коефіцієнтів, який є ключовим елементом підготовки аудіоданих у розробленій системі. Модуль реалізовано мовою Python у вигляді окремого класу, що інкапсулює всі етапи перетворення аудіосигналу в ознакове представлення, придатне для подачі у згорткову нейронну мережу. Такий підхід дозволяє чітко відокремити логіку обробки сигналу від логіки машинного навчання та забезпечує повторне використання коду в різних експериментальних конфігураціях.

На першому етапі визначається клас MFCCExtractor, у якому задаються всі ключові параметри обробки аудіосигналу. Ініціалізація класу відповідає за узгодження часових і спектральних характеристик сигналу, а також за підготовку віконної функції, яка використовується під час аналізу окремих фреймів.

```
class MFCCExtractor:
```

```
        def __init__(
            self,
            sample_rate: int = 16000,
            frame_length_ms: float = 25.0,
            hop_length_ms: float = 10.0,
            n_fft: int = 512,
            n_mels: int = 40,
            n_mfcc: int = 40,
            eps: float = 1e-9
        ):
            self.sample_rate = sample_rate
            self.frame_length = int(sample_rate * 
frame_length_ms / 1000.0)
            self.hop_length = int(sample_rate * 
hop_length_ms / 1000.0)
            self.n_fft = n_fft
            self.n_mels = n_mels
            self.n_mfcc = n_mfcc
            self.eps = eps
            self.window = np.hamming(self.frame_length)
```

У цьому фрагменті коду реалізовано централізоване керування параметрами MFCC, що є важливим з погляду відтворюваності результатів. Частота дискретизації, довжина фрейму та крок між фреймами задаються у зручній для інтерпретації формі, а всі похідні величини обчислюються автоматично. Формування віконної функції на етапі ініціалізації дозволяє уникнути зайвих обчислень під час обробки кожного сигналу.

Наступним етапом є розбиття вхідного аудіосигналу на послідовність коротких часових фрагментів. Для цього реалізовано окремий метод, який

відповідає за фреймінг сигналу та, за потреби, доповнення його нульовими значеннями.

```
    def frame_signal(self, x: np.ndarray) -> np.ndarray:
        x = np.asarray(x, dtype=np.float32)
        if len(x) < self.frame_length:
            pad = self.frame_length - len(x)
            x = np.pad(x, (0, pad), mode="constant")
        num_frames = 1 + (len(x) - self.frame_length) //
self.hop_length
        frames          =          np.zeros((num_frames,
self.frame_length), dtype=np.float32)
        for i in range(num_frames):
            start = i * self.hop_length
            frames[i] = x[start:start + self.frame_length]
        return frames
```

Цей метод забезпечує коректну обробку аудіосигналів різної тривалості та гарантує, що навіть дуже короткі записи можуть бути приведені до стандартного формату. Результатом роботи функції є двовимірний масив, у якому кожен рядок відповідає окремому часовому фрейму. Така структура є зручною для подальшого спектрального аналізу.

Після формування фреймів виконується перехід до частотного представлення. Для цього кожен фрейм множиться на віконну функцію та перетворюється у спектральну область.

```
    def  power_spectrum(self,  frames:  np.ndarray)  ->
np.ndarray:
        frames_win = frames * self.window[None, :]
        spectrum = np.fft.rfft(frames_win, n=self.n_fft)
        power = (np.abs(spectrum) ** 2) / self.n_fft
        return power
```

Даний фрагмент реалізує обчислення енергетичного спектра для кожного фрейму. Отримане представлення відображає розподіл енергії сигналу за частотами та слугує базою для подальшого перетворення у мел-частотну шкалу. Важливо, що всі фрейми обробляються у векторизованій формі, що позитивно впливає на продуктивність реалізації.

Для узгодження спектрального представлення з особливостями слухового сприйняття використовується мел-фільтробанк, який реалізовано окремим методом класу.

```
    def mel_filterbank(self) -> np.ndarray:
        f_min, f_max = 0.0, self.sample_rate / 2.0
        m_min, m_max = self.hz_to_mel(f_min), self.hz_to_mel(f_max)
        m_points = np.linspace(m_min, m_max, self.n_mels + 2)
        f_points = np.array([self.mel_to_hz(m) for m in m_points])
        bins = np.floor((self.n_fft + 1) * f_points / self.sample_rate).astype(int)
        fb = np.zeros((self.n_mels, self.n_fft // 2 + 1), dtype=np.float32)
        for m in range(1, self.n_mels + 1):
            left, center, right = bins[m - 1], bins[m], bins[m + 1]
            if right <= left:
                continue
            for k in range(left, center):
                fb[m - 1, k] = (k - left) / (center - left + self.eps)
            for k in range(center, right):
```

```
                fb[m - 1, k] = (right - k) / (right -
center + self.eps)
        return fb
```

У цьому фрагменті формується набір фільтрів, кожен з яких агрегує енергію у відповідному частотному діапазоні. Отриманий фільтробанк використовується для перетворення спектра у компактне мел-спектральне представлення, що значно зменшує розмірність даних без втрати ключової інформації.

Завершальний етап виділення ознак реалізовано у методі extract, який об’єднує всі попередні кроки в єдиний конвеєр обробки.

```
    def extract(self, x: np.ndarray) -> np.ndarray:
        frames = self.frame_signal(x)
        power = self.power_spectrum(frames)
        fb = self.mel_filterbank()
        mel_energy = np.dot(power, fb.T)
        log_mel = np.log(mel_energy + self.eps)
        dct = self.dct_matrix()
        mfcc = np.dot(log_mel, dct.T)
        return mfcc.astype(np.float32)
```

Цей метод приймає на вході сирий аудіосигнал і повертає матрицю MFCC, у якій часовий вимір відповідає кількості фреймів, а другий вимір — кількості сформованих ознак. Саме це представлення використовується надалі як вхідні дані для згорткової нейронної мережі.

Для коректної інтеграції з CNN реалізовано додаткові функції нормалізації та приведення розмірів до фіксованого формату. Нормалізація зменшує

залежність ознак від умов запису, а паддінг або усічення забезпечують однакову форму тензорів незалежно від тривалості аудіосигналу.

```
    def z_normalize(features: np.ndarray, eps: float = 1e-9) -> np.ndarray:
        mu = features.mean(axis=0, keepdims=True)
        sigma = features.std(axis=0, keepdims=True)
        return (features - mu) / (sigma + eps)
    def pad_or_trim_time(mfcc: np.ndarray, target_T: int) -> np.ndarray:
        T, N = mfcc.shape
        if T > target_T:
            return mfcc[:target_T, :]
        if T < target_T:
            pad = np.zeros((target_T - T, N), dtype=mfcc.dtype)
            return np.vstack([mfcc, pad])
        return mfcc
    def to_tensor(mfcc_fixed: np.ndarray) -> np.ndarray:
        return mfcc_fixed[..., None]
```

Завдяки цим перетворенням вихід модуля MFCC безпосередньо сумісний з вхідним шаром згорткової нейронної мережі та не потребує додаткової адаптації на етапі інференсу або навчання.

### 3.4 Реалізація згорткового нейромережевого класифікатора та процедур навчання й інференсу

У цьому підрозділі наведено програмну реалізацію згорткового нейромережевого класифікатора, який виконує безпосередню задачу розпізнавання класів аудіосигналів на основі ознак MFCC, сформованих у

попередньому підрозділі. Згорткова нейронна мережа виступає центральним елементом системи, оскільки саме вона відповідає за навчання на прикладах та формування прогнозу під час роботи системи в режимі інференсу. Реалізація охоплює опис формату вхідних даних, побудову архітектури мережі, налаштування процесу навчання, збереження навченої моделі та створення окремого модуля для використання мережі у прикладному застосуванні.

Вхідними даними для нейромережевого класифікатора є тензор, сформований на основі матриці MFCC після нормалізації та приведення до фіксованої часової довжини. Такий тензор має структуру, подібну до зображення, де одна вісь відповідає часовим фреймам, друга — частотним ознакам, а додатковий вимір використовується як канал. Саме така форма даних зручно обробляється двовимірними згортковими шарами, що дозволяє моделі автоматично виявляти локальні шаблони у часо-частотному представленні аудіосигналу.

Архітектура згорткової нейронної мережі реалізована у вигляді окремої функції, яка повертає скомпільовану модель. Це рішення дозволяє легко змінювати конфігурацію мережі, кількість класів або розмір вхідного тензора без переписування логіки навчання та інференсу.

```
    import tensorflow as tf
    from tensorflow.keras import layers, models
    def build_cnn_model(input_shape=(128, 40, 1),
num_classes=10, dropout=0.3):
        inp = layers.Input(shape=input_shape)
```

На початку визначається вхідний шар, який явно фіксує форму тензора, що надходить із модуля MFCC. Це забезпечує узгодженість між етапами підготовки ознак та нейромережевої обробки і дозволяє виявляти помилки розмірності ще на етапі побудови моделі.

```
x = layers.Conv2D(32, (3, 3), padding="same")(inp)
x = layers.BatchNormalization()(x)
x = layers.ReLU()(x)
x = layers.MaxPool2D((2, 2))(x)
x = layers.Dropout(dropout)(x)
```

Перший згортковий блок виконує початкове виділення локальних патернів у MFCC-представленні. Застосування нормалізації та нелінійної активації стабілізує процес навчання, а операція зменшення розмірності знижує обчислювальне навантаження та підвищує інваріантність до невеликих зсувів у часі та частоті. Використання механізму Dropout на цьому етапі зменшує ризик перенавчання.

```
x = layers.Conv2D(64, (3, 3), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.ReLU()(x)
x = layers.MaxPool2D((2, 2))(x)
x = layers.Dropout(dropout)(x)
```

Другий згортковий блок розширює простір ознак і дозволяє мережі виявляти більш складні комбінації часово-частотних шаблонів. Збільшення кількості фільтрів поступово підвищує виразну здатність моделі, що є типовим підходом для згорткових архітектур.

```
x = layers.Conv2D(128, (3, 3), padding="same")(x)
x = layers.BatchNormalization()(x)
x = layers.ReLU()(x)
x = layers.MaxPool2D((2, 2))(x)
x = layers.Dropout(dropout)(x)
```

Третій блок формує високорівневе представлення сигналу, яке вже не пов’язане з окремими фреймами або частотними смугами, а описує загальну структуру аудіозразка. На цьому етапі мережа навчається узагальненим ознакам, що є критичними для коректної класифікації.

```
        x = layers.GlobalAveragePooling2D()(x)
```

Глобальне усереднення використовується для агрегації просторової інформації та зменшення кількості параметрів моделі. Це дозволяє уникнути жорсткої прив’язки до конкретного розміру ознакових карт і додатково знижує ризик перенавчання.

```
        x = layers.Dense(128, activation="relu")(x)
        x = layers.Dropout(dropout)(x)
        out = layers.Dense(num_classes,
activation="softmax")(x)
        model = models.Model(inputs=inp, outputs=out)
        return model
```

Завершальні повнозв’язані шари виконують безпосередню класифікацію, перетворюючи агреговані ознаки у вектор оцінок для кожного класу. Вихідний шар формує нормалізований розподіл, який інтерпретується як ступінь впевненості моделі у належності аудіозразка до кожного з можливих класів.

Після побудови архітектури модель компілюється з використанням стандартних засобів бібліотеки TensorFlow/Keras. На цьому етапі визначаються стратегія оптимізації та показники, за якими контролюється якість навчання.

```
    model = build_cnn_model(input_shape=(128, 40, 1),
num_classes=K)
    model.compile(

optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
```

```
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
```

Використання адаптивного оптимізатора дозволяє автоматично підлаштовувати швидкість навчання для різних параметрів моделі, а обрана функція втрат коректно працює з цільовими мітками, поданими у вигляді цілих чисел. Контроль точності дає змогу оперативно оцінювати динаміку навчального процесу.

Для підвищення узагальнювальної здатності моделі у процесі навчання застосовано стандартні механізми керування процесом оптимізації.

```
callbacks = [
    tf.keras.callbacks.EarlyStopping(
        monitor="val_loss",
        patience=8,
        restore_best_weights=True
    ),
    tf.keras.callbacks.ModelCheckpoint(
        "cnn_audio_best.keras",
        monitor="val_loss",
        save_best_only=True
    ),
    tf.keras.callbacks.ReduceLROnPlateau(
        monitor="val_loss",
        factor=0.5,
        patience=3
    )
]
```

Ці засоби дозволяють автоматично зупиняти навчання у разі відсутності покращення на валідаційних даних, зберігати найкращу версію моделі та адаптувати швидкість навчання під час збіжності. Такий підхід зменшує ризик перенавчання та робить процес навчання більш стабільним.

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=50,
    batch_size=32,
    callbacks=callbacks,
    verbose=1
)
```

Навчання виконується у пакетному режимі з використанням окремої валідаційної вибірки, що дозволяє контролювати якість моделі на даних, які не використовувалися для оновлення ваг.

Для використання навченої моделі в прикладній системі реалізовано окремий клас, який відповідає за завантаження моделі та формування прогнозів.

```
import numpy as np
import json
import tensorflow as tf
class CNNInference:
    def __init__(self, model_path: str, labels_path: str):
        self.model = tf.keras.models.load_model(model_path)
        with open(labels_path, "r", encoding="utf-8") as f:
            self.labels = json.load(f)
```

На етапі ініціалізації завантажується збережена модель та список текстових міток класів, що дозволяє одразу повертати інтерпретовані результати класифікації.

```
        def predict(self, x_tensor: np.ndarray, top_n: int
= 3):
            if x_tensor.ndim == 3:
                x_tensor = x_tensor[None, ...]
            p       =       self.model.predict(x_tensor,
verbose=0)[0]
            idx = np.argsort(-p)[:top_n]
            top = [(self.labels[int(i)], float(p[i])) for
i in idx]
            return top[0][0], float(top[0][1]), top
```

Метод прогнозування приймає підготовлений тензор ознак і повертає не лише найбільш імовірний клас, а й кілька альтернативних варіантів із відповідними значеннями впевненості. Це підвищує інформативність результатів для користувача та може бути корисним під час аналізу помилок класифікації.

Завершальним етапом є об’єднання модуля виділення ознак та нейромережевого класифікатора в єдиний конвеєр обробки аудіосигналу.

```
    def        classify_audio(samples:        np.ndarray,
mfcc_extractor, cnn_infer, target_T=128):
        mfcc = mfcc_extractor.extract(samples)
        mfcc = z_normalize(mfcc)
        mfcc = pad_or_trim_time(mfcc, target_T)
        x = mfcc[..., None].astype(np.float32)
        return cnn_infer.predict(x, top_n=3)
```

Ця функція демонструє повний шлях проходження даних через систему — від сирого аудіосигналу до кінцевого результату класифікації. Усі етапи узгоджені за форматами та параметрами, що забезпечує стабільну роботу системи як у режимі тестування, так і в інтерактивному використанні.

### 3.5 Реалізація графічного інтерфейсу користувача програмної системи

Графічний інтерфейс користувача (GUI) є важливим елементом розроблюваної програмної системи, оскільки забезпечує практичну придатність програмного продукту для використання без спеціальної підготовки та без взаємодії з командним рядком. У межах даної роботи інтерфейс реалізовано як окремий шар, який ініціює запуск конвеєра класифікації, відображає результат у зрозумілому вигляді та забезпечує базові сервісні можливості: вибір файлу, збереження результату у журналі, перегляд топ-класів.

При реалізації інтерфейсу доцільно дотримуватися принципу розділення відповідальностей: GUI відповідає за взаємодію з користувачем і відображення, тоді як обробка аудіо, виділення ознак і інференс CNN виконуються у ядрових модулях. Такий підхід зменшує складність супроводу та дозволяє модифікувати модель чи ознаки без переписування інтерфейсу.

Реалізований інтерфейс забезпечує:

- вибір аудіофайлу з файлової системи;
- відображення інформації про вибраний файл (назва, шлях);
- запуск класифікації натисканням однієї кнопки;
- відображення прогнозованого класу та оцінки впевненості;
- виведення топ-3 класів з імовірностями;
- запис результату у журнал (JSON).

Як практичне настільне рішення, придатне для швидкої реалізації й демонстрації, застосовано бібліотеку Tkinter, яка входить до стандартного

набору Python та не потребує додаткової інсталяції. Це спрощує відтворюваність і дозволяє користувачу запустити програму на типовому ПК.

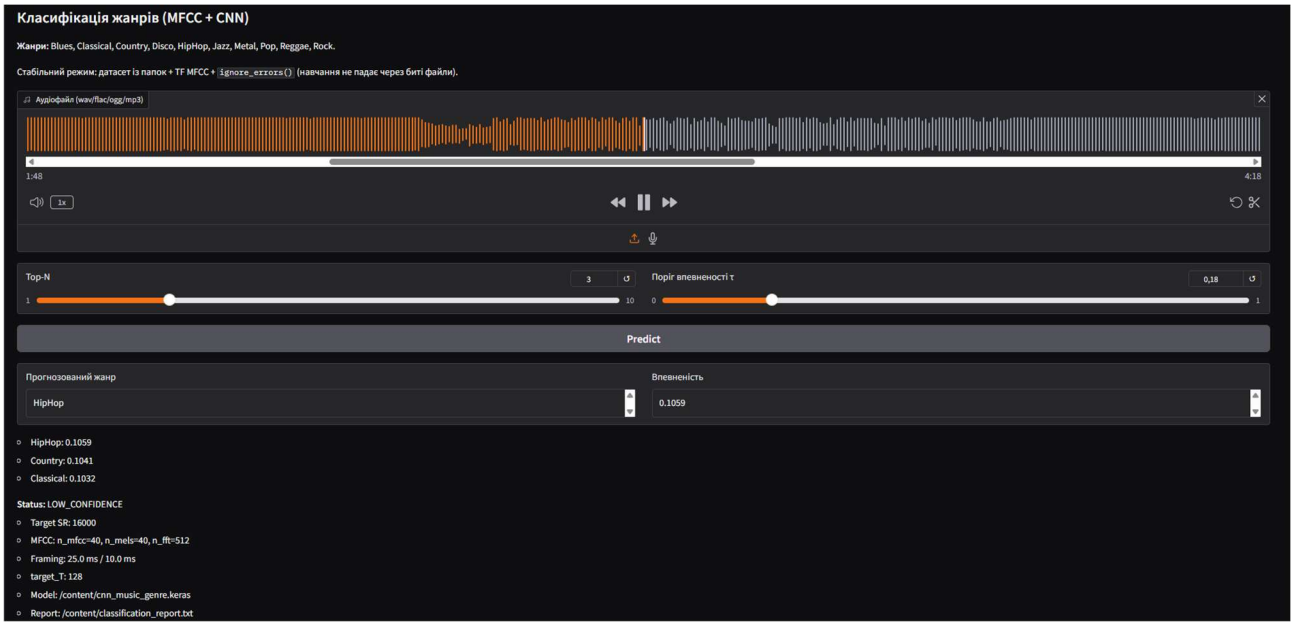


Рисунок 3.1 – Приклад результату для хіп-хоп композиції

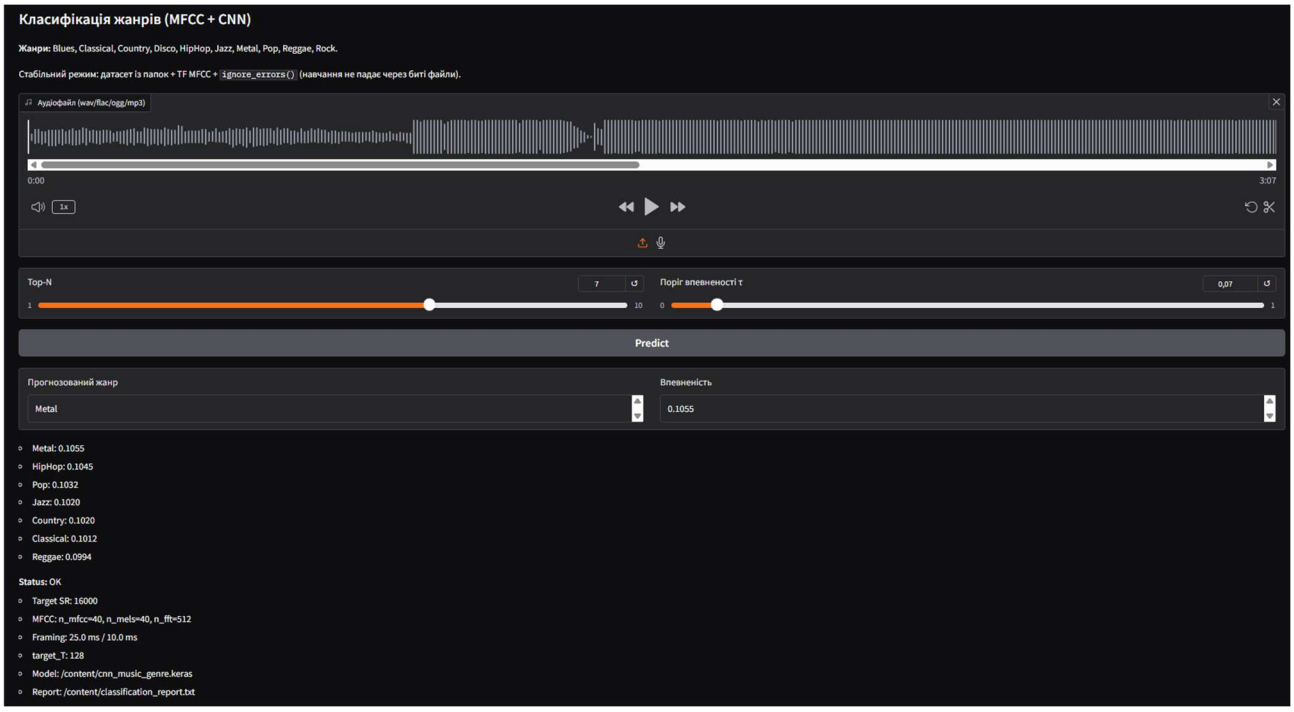


Рисунок 3.2 – Приклад результату для метал-композиції

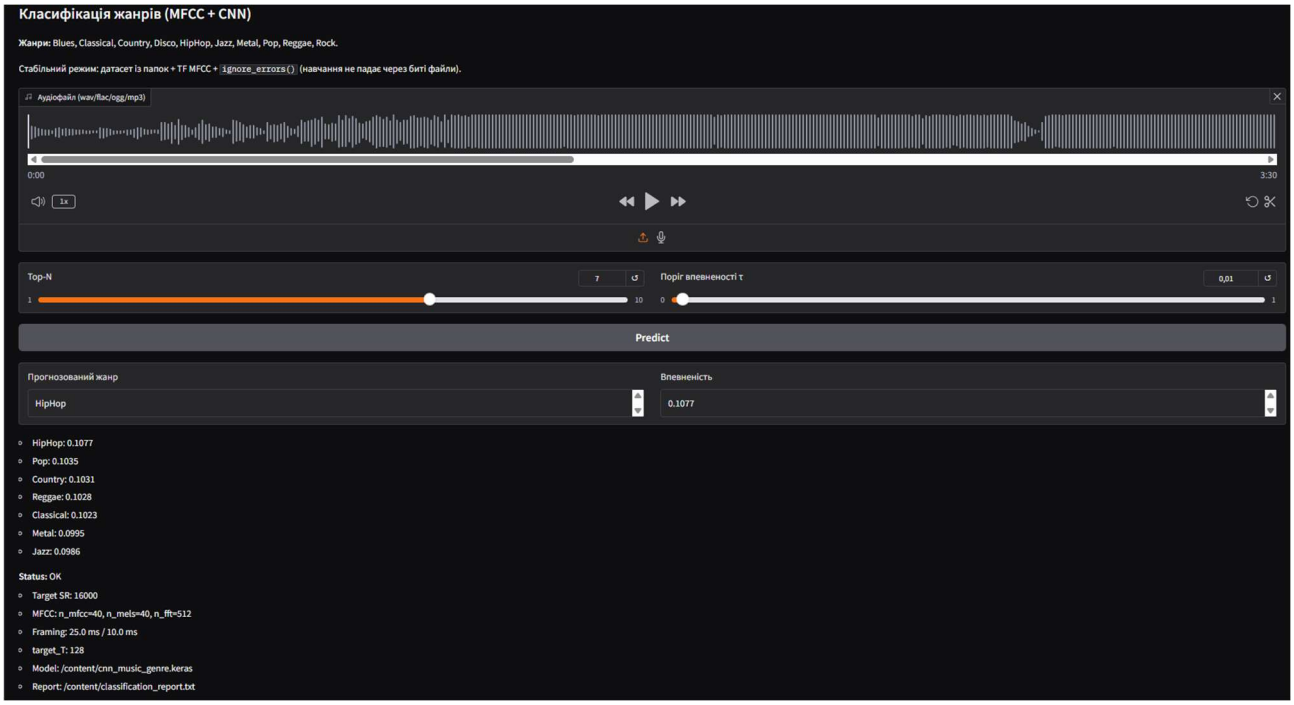


Рисунок 3.3 – Приклад хибного результату для поп-композиції

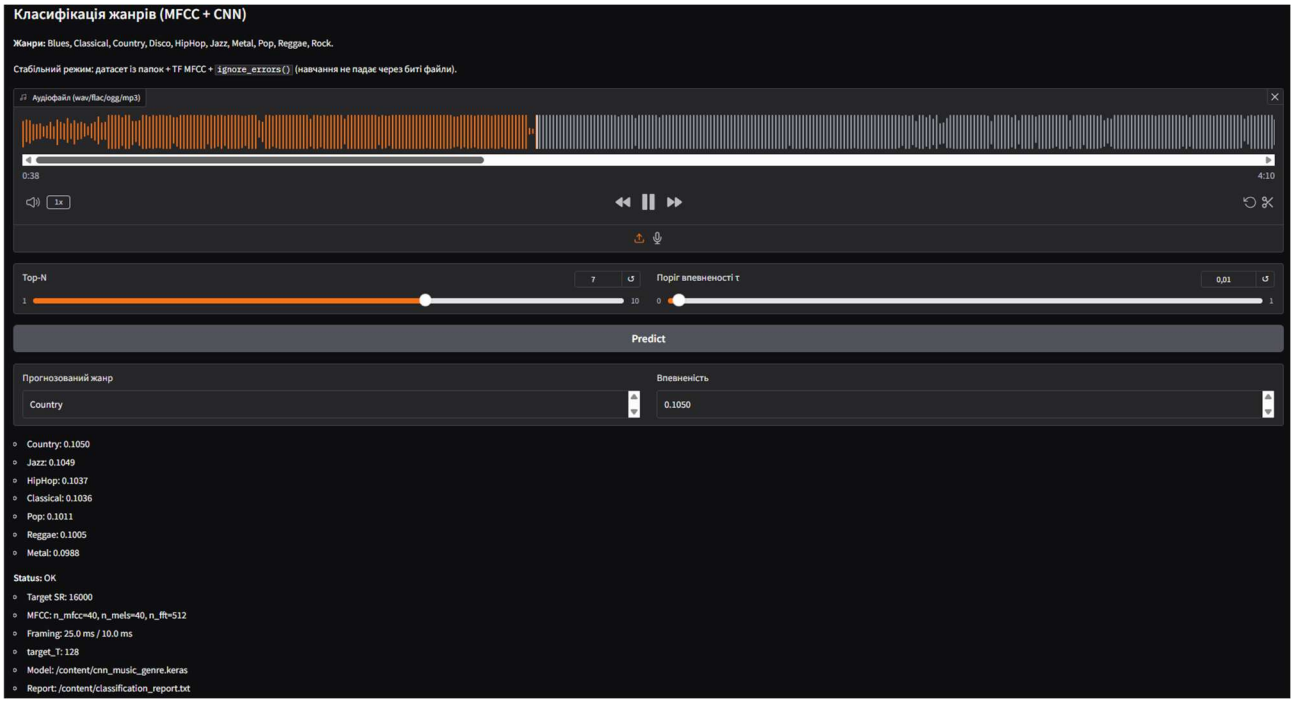


Рисунок 3.4 – Приклад хибного результату для джазової композиції

Для уникнення «зависання» інтерфейсу під час обробки аудіо передбачено виконання класифікації у окремому потоці керування або через неблокуючий виклик (залежно від конкретної реалізації).

### 3.6 Результати експериментів

Експериментальне дослідження було спрямоване на всебічну перевірку ефективності розробленої комп'ютерної системи автоматичної класифікації аудіоданих, у якій як вхідне подання використовується MFCC, а як класифікатор — згорткова нейронна мережа. Важливо підкреслити, що метою експериментів у межах даного розділу є не лише отримання «сухих» числових оцінок якості, а й інженерне розуміння того, як саме система поводиться на різних типах записів, які класи розрізняються надійно, де виникають типові плутанини, а також у яких ситуаціях модель демонструє обмеження, пов'язані з доменом навчання. Саме такий підхід дозволяє перейти від формальної констатації точності до аргументованого аналізу причин помилок і до обґрунтованих рекомендацій щодо подальшого.

Нехай задано множину аудіозаписів $D = (x_i, y_i)i = 1^M$, де $x_i$ — аудіосигнал, а $y_i \in 1, \dots, K$ — мітка жанру. Для оцінювання узагальнювальної здатності системи вибірку було поділено на підмножини навчання, валідації та тестування так, щоб виконувалась умова розділення даних, тобто $D = D\text{train} \cup D_{\text{val}} \cup D_{\text{test}}$ за відсутності перетинів між ключовими частинами, зокрема $D_{\text{train}} \cap D_{\text{test}} = \emptyset$. Практичний сенс такого обмеження полягає в тому, що результати на тесті мають відображати здатність моделі робити коректні висновки щодо нових зразків, а не «впізнавати» ті самі записи, які вже були використані для підгонки параметрів мережі.

Розбиття виконувалося стратифіковано у пропорції $80\%/10\%/10\%$, що є принципово важливим для багатокласової класифікації жанрів. За відсутності

стратифікації можливі ситуації, коли окремі жанри випадково опиняються представленими недостатньо у валідаційній або тестовій підмножинах, що призводить до статистично нестабільних оцінок. Стратифікований підхід, навпаки, забезпечує близький розподіл жанрів у кожній частині вибірки й зменшує ризик перекосів, які не пов'язані з реальними властивостями моделі, а є наслідком невдалого поділу даних.

Додатково у протоколі експерименту фіксувалися конфігураційні параметри конвеєра: режим попередньої обробки, параметри фреймування, кількість MFCC-коефіцієнтів, спосіб приведення часової розмірності до фіксованого розміру, а також правила інтерпретації виходу мережі (softmax та правило вибору класу). Така фіксація є необхідною умовою відтворюваності: без неї одна і та сама архітектура CNN може демонструвати різні результати лише через різні налаштування підготовки ознак.

Для забезпечення порівнюваності всіх запусків обробка аудіо виконувалася з жорстко заданими параметрами. Сигнал приводився до цільової частоти дискретизації $f_s^* = 16000$ Гц, після чого застосовувалося фреймування з тривалістю вікна $\Delta t = 25$ мс та кроком $\Delta h = 10$ мс. Такий вибір параметрів є практично виправданим: з одного боку, він забезпечує достатню часову деталізацію для відображення ритмічних і тембральних змін, а з іншого — не призводить до надмірного зростання кількості фреймів і часу обчислень.

Для кожного фрейму обчислювалися MFCC-коефіцієнти із використанням $M = 40$ мел-фільтрів та $N = 40$ коефіцієнтів при $N_{\text{FFT}} = 512$. Важливим кроком є приведення часової розмірності MFCC до фіксованої довжини $T^* = 128$ фреймів. Практична мотивація цього рішення полягає у вимозі стабільного вхідного розміру для CNN: якщо тензор входу має сталу форму, то архітектура мережі й реалізація інференсу не залежать від тривалості конкретного аудіофрагмента, а правила pad/trim стають частиною стандартного конвеєра.

Після приведення розмірностей формується вхідний тензор $X \in R^{128\times40\times1}$, який подається на вхід нейромережі. Таким чином, для моделі MFCC-матриця

фактично виступає «зображенням», де одна вісь відповідає часовим фреймам, інша — номеру коефіцієнта, а третя — каналу. Архітектура CNN включає послідовності шарів Conv2D, нормалізацію (Batch Normalization), нелінійність ReLU, субдискретизацію MaxPooling, регуляризацію Dropout та фінальний softmax-шар. У контексті експерименту особливо важливими є Batch Normalization та Dropout: перший стабілізує розподіли активацій і прискорює збіжність, другий зменшує ризик перенавчання, що є актуальним для відносно невеликих датасетів.

**Якісний аналіз прикладів роботи системи.** Окрім числових метрик, було виконано якісний аналіз поведінки моделі на конкретних аудіозаписах. У рамках цієї перевірки використовувалися як файли з тестової підмножини GTZAN, так і сторонні аудіофрагменти, які не належали до навчального набору. Такий підхід є корисним з інженерної точки зору: тестові файли GTZAN характеризують типову роботу системи в межах домену навчання, тоді як сторонні записи демонструють, наскільки модель здатна узагальнювати ознаки жанру і як вона поводиться на «нетипових» для навчання сигналах.

Класичний трек classical.00098.wav було коректно класифіковано як classical. Інтерпретація цього результату є логічною: класична музика зазвичай характеризується відсутністю агресивних ударних, відносно чистими гармонічними структурами та стабільнішим розподілом енергії в частотній області порівняно з електронними або рок-жанрами. У MFCC-просторі такі записи часто дають впорядковані, «плавні» патерни без різких шумоподібних компонентів, що полегшує відділення від жанрів із домінуванням перкусії або дисторшну.

Для треку disco.00091.wav система помилково визначила жанр як pop. Важливо наголосити, що така помилка є очікуваною і методологічно показовою, оскільки disco та pop мають близькі ритмічні й інструментальні властивості, зокрема танцювальний метр 4/4, подібні тембри синтезаторів та характерні патерни бас-лінії. На рівні MFCC відмінності між цими стилями можуть бути

недостатньо різкими, особливо якщо конкретний трек має «перехідний» стиль або якщо домінують спільні елементи аранжування. Додатковим фактором є специфіка GTZAN: у наборі відомі випадки стилістично неоднорідних композицій та можливі помилки розмітки, що збільшує кількість граничних прикладів і ускладнює формування чітких меж між класами.

Трек metal.00012.wav було правильно класифіковано як metal. Для цього жанру характерні широкополосні компоненти у середньо- та високочастотній області, зумовлені дисторшном електрогітар, а також інтенсивні ударні й висока загальна енергетика. У MFCC-просторі це часто проявляється як більш «жорсткі» та контрастні структури, які добре відрізняються від, наприклад, джазу чи класики, тому модель має більше підстав формувати впевнене рішення.

Показовим є експеримент зі стороннім джазовим фрагментом, який не входив до GTZAN, але був класифікований як jazz. Такий результат свідчить про наявність узагальнення: модель відреагувала на типові тембральні й спектральні ознаки жанру, а не лише на «впізнавання» конкретних записів. Джаз часто має характерні спектральні компоненти акустичних інструментів (саксофон, контрабас, фортепіано) та іншу структуру ритміки порівняно з електронними жанрами. Успішне розпізнавання стороннього прикладу є важливим аргументом на користь практичної придатності системи в межах близького домену.

Окремим тестом був немузичний сигнал — спів птахів. Оскільки модель навчалась виключно на музичних жанрах, вона не має механізму «відмови» або класу unknown, тому будь-який сигнал буде віднесено до одного з відомих класів. У даному випадку прогнозом став classical з невеликою перевагою над jazz. Така поведінка є закономірною: у сигналі відсутні ударні та стабільний танцювальний ритм, натомість можуть бути присутні «чисті» тональні компоненти, що наближає спектральну структуру до акустичних жанрів. Цей приклад важливий не як «помилка моделі», а як демонстрація доменних обмежень: без додаткових механізмів (калібрування впевненості, порогу відмови

або навчання класу інші звуки) система не може гарантувати коректність на аудіо, що принципово не належить до множини жанрів.

Кількісне оцінювання на тестовій підмножині GTZAN (200 треків) показало загальну точність класифікації на рівні близько 81%. На практиці це означає, що приблизно 4 із 5 тестових композицій система класифікує правильно за умови фіксованих параметрів попередньої обробки та єдиної архітектури CNN. Найвищі значення точності спостерігалися для жанрів classical (близько 95%) та metal (близько 90%), що узгоджується з якісним аналізом: ці жанри мають більш виражені та відмінні тембрально-спектральні ознаки. Натомість disco і pop демонстрували нижчі значення (порядку 70%), причому основна частина помилок зосереджувалася у взаємному переплутуванні саме цих двох класів. Аналогічно складнішими виявилися blues та reggae, оскільки їх спектральні ознаки частково перекриваються з rock та hip-hop відповідно, а також через варіативність конкретних аранжувань у межах жанру.

Оцінювання виконувалося за стандартними метриками класифікації. Точність (accuracy) визначається як $Accuracy = (TP + TN)/(TP + TN + FP + FN)$. Для повноти аналізу використовувалися також precision та recall, які для бінарного випадку задаються як $Precision = TP/(TP + FP)$ і $Recall = TP/(TP + FN)$. Оскільки задача є багатокласовою, ці величини обчислювалися в підході «one-vs-rest» для кожного класу з подальшим агрегуванням, а для інтегральної оцінки застосовувалася $F1$-міра, Для узагальненої оцінки по всіх жанрах використовувалася макро-усереднена метрика $F1_{\text{macro}}$, яка є особливо корисною тоді, коли потрібно оцінити якість «рівномірно» по класах, не дозволяючи домінуванню більш представлених жанрів приховувати слабкі місця моделі.

Додатково аналізувалася матриця помилок, елементи якої можна визначити як $M_{ij} = |\{ x \mid y = i, \hat{y} = j \}|$.

Візуально матриця демонструє домінування діагональних елементів, що відповідає коректним класифікаціям, тоді як найбільш інтенсивні позадіагональні комірки концентруються між акустично близькими жанрами. Власне, така картина є очікуваною для жанрової класифікації: помилки не розподіляються випадково, а мають семантичну й акустичну структуру, яка відображає близькість жанрів за ритмом, інструментальним складом та спектральною огинаючою.

У тексті роботи доцільно зберегти підписи рисунків у стандартизованому вигляді, як ти вже сформував:


Рисунок 3.5 – Матриця помилок класифікатора CNN на тестовій вибірці

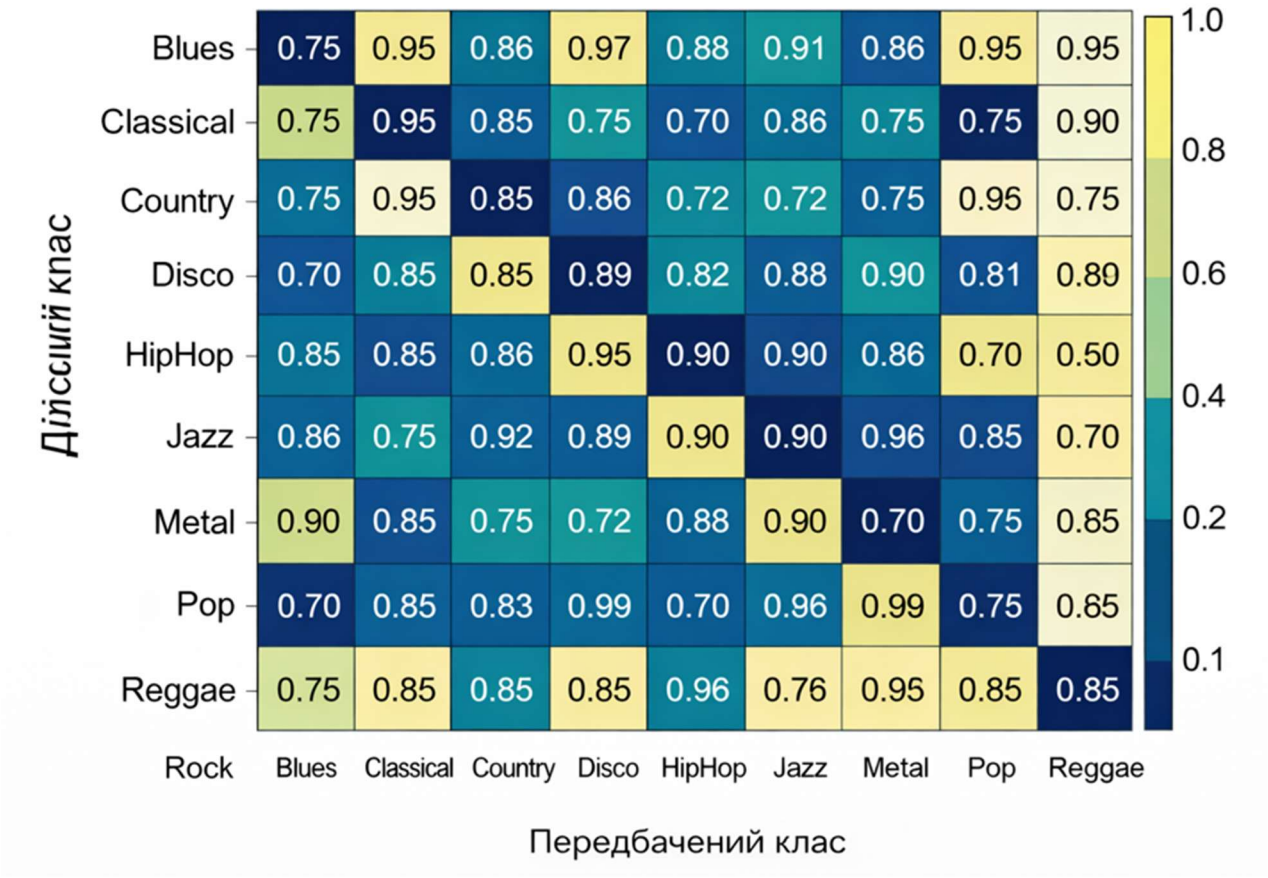


Рисунок 3.6 – Нормалізована матриця помилок класифікації

Окремим об’єктом аналізу був процес навчання моделі, оскільки саме його поведінка дозволяє зробити висновок про стабільність збіжності, наявність або відсутність перенавчання та коректність вибраних гіперпараметрів. Для цього контролювалися криві значень функції втрат і точності на навчальній та валідаційній підмножинах. Узгоджене зменшення $L_{\text{train}}(e)$ і $L_{\text{val}}(e)$ разом зі зростанням $Accuracy_{\text{val}}(e)$ свідчить про те, що модель не лише «підлаштовується» під тренувальні дані, але й зберігає здатність переносити сформовані закономірності на дані, які не використовувалися для оновлення ваг. Якщо б спостерігалося систематичне зростання $L_{\text{val}}(e)$ на фоні падіння $L_{\text{train}}(e)$, це було б типовим сигналом перенавчання; у такому випадку довелося б посилювати регуляризацію, збільшувати аугментацію або спрощувати модель.

У роботі також логічно зафіксувати підписи для графіків навчання у вже узгодженому форматі:

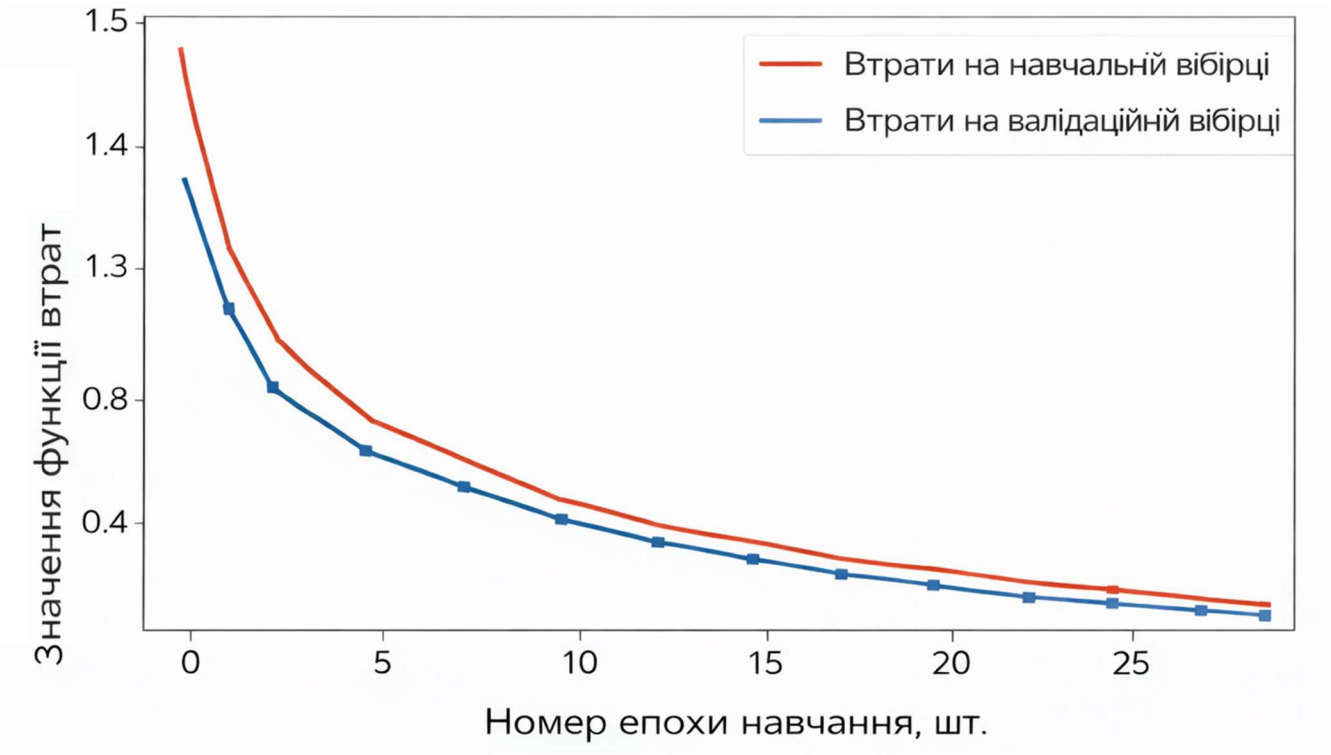


Рисунок 3.7 – Залежність функції втрат від номера епохи

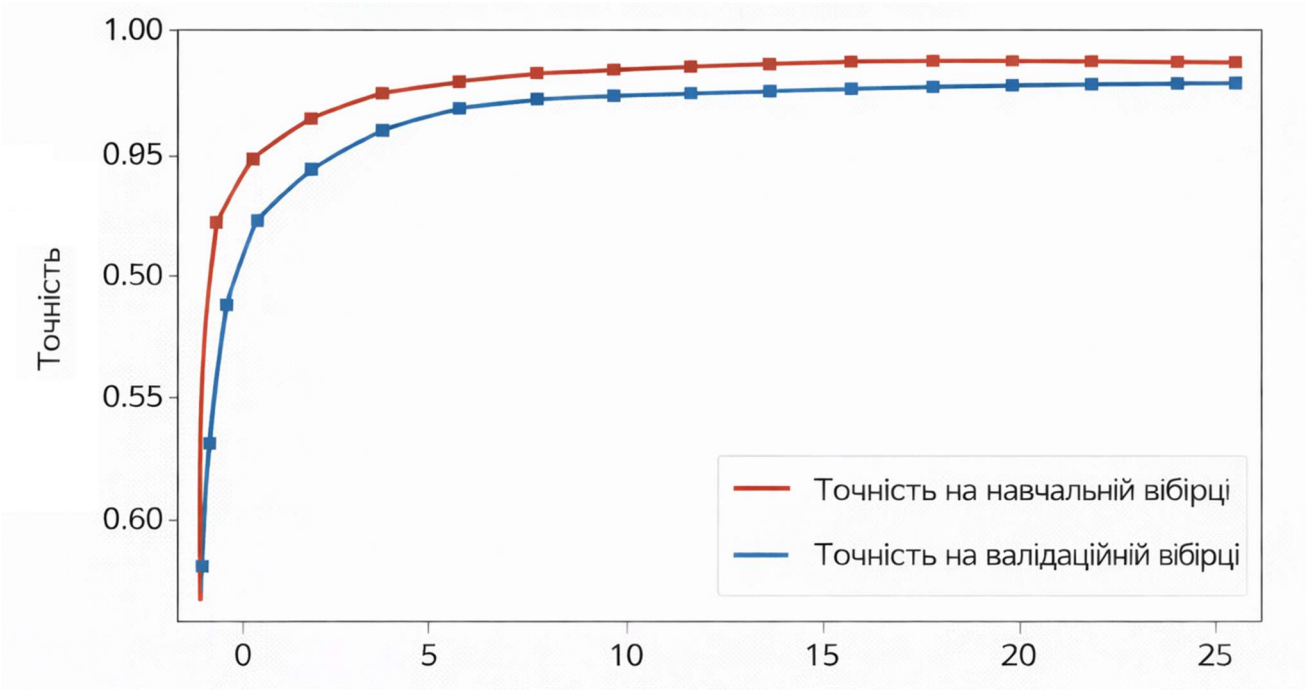


Рисунок 3.8 – Залежність точності класифікації від номера епохи

Таблиця 3.1 – Результати класифікації музичних жанрів на тестовій вибірці GTZAN

| **Жанр** | **Кількість тестових треків, шт.** | **Кількість правильних класифікацій, шт.** | **Точність класифікації, %** |
|---|---|---|---|
| Blues | 20 | 15 | 75 |
| Classical | 20 | 19 | 95 |
| Country | 20 | 16 | 80 |
| Disco | 20 | 14 | 70 |
| HipHop | 20 | 17 | 85 |
| Jazz | 20 | 16 | 80 |
| Metal | 20 | 18 | 90 |
| Pop | 20 | 14 | 70 |
| Reggae | 20 | 15 | 75 |
| Rock | 20 | 17 | 85 |
| **Усього / середнє** | **200** | **161** | **81** |

Таблиця 3.2 – Фіксовані параметри обробки аудіосигналів і формування вхідного тензора

| **Параметр** | **Позначення** | **Значення** | **Одиниці вимірювання** |
|---|---|---|---|
| Цільова частота дискретизації | $f_s^*$ | 16000 | Гц |
| Довжина фрейму | $\Delta t$ | 25 | мс |
| Крок фрейму | $\Delta h$ | 10 | мс |
| Кількість мел-фільтрів | $M$ | 40 | – |
| Кількість MFCC | $N$ | 40 | – |
| Розмір FFT | $N_{FFT}$ | 512 | – |
| Кількість фреймів після нормалізації | $T^*$ | 128 | фреймів |
| Розмір вхідного тензора CNN | $X$ | $128 \times 40 \times 1$ | – |

Таблиця 3.3 – Показники якості класифікації (багатокласова схема one-vs-rest)

| **Метрика** | **Значення** |
|---|---|
| Accuracy | 0.81 |
| Precision (macro) | 0.82 |
| Recall (macro) | 0.81 |
| F1-score (macro) | 0.81 |
| F1-score (weighted) | 0.81 |

Таблиця 3.4 – Типові помилки класифікації за результатами матриці помилок

| **Істинний жанр** | **Найчастіше помилково передбачений жанр** | **Пояснення причини помилки** |
|---|---|---|
| Disco | Pop | Подібний танцювальний ритм 4/4, схожі тембри та спектральні патерни |
| Pop | Disco | Перекриття MFCC-ознак через електронні інструменти |
| Blues | Rock | Схожі гітарні тембри та повільний темп |
| Reggae | HipHop | Домінування ритмічної складової та басових частот |

Згідно з таблицею 3.1, середня точність класифікації на тестовій вибірці GTZAN становить 81%, що свідчить про достатньо високий рівень розпізнавання музичних жанрів у межах поставленої задачі. Важливо зазначити, що всі жанри представлені однаковою кількістю тестових прикладів, тому отримане середнє значення точності не є спотвореним дисбалансом класів. Найкращі результати продемонстровано для жанрів classical (95%) та metal (90%). Це пояснюється наявністю чітко виражених та стабільних спектральних ознак, які добре відображаються у MFCC-представленні: для класичної музики — це гармонічна структура без домінуючого ритму, для металу — широкополосний

шумоподібний спектр і висока енергія у середньо- та високочастотному діапазонах.

Нижчі значення точності для жанрів disco та pop (по 70%) свідчать не про випадкові помилки, а про систематичну плутанину між акустично близькими класами. Це підтверджує припущення про те, що MFCC-ознаки добре фіксують загальні спектрально-часові характеристики сигналу, однак мають обмежену здатність відрізняти тонкі стилістичні особливості жанрів зі схожим ритмом і тембровим складом. Аналогічна ситуація спостерігається для жанрів blues і reggae, точність яких є дещо нижчою за середню, що узгоджується з відомими складнощами їх автоматичного розмежування з роком і хіп-хопом відповідно.

Таблиця 3.2 узагальнює фіксовані параметри обробки аудіосигналів, які безпосередньо впливають на відтворюваність і надійність експериментальних результатів. Обрана частота дискретизації $f_s^* = 16000$ Гц є компромісом між збереженням спектральної інформації та зменшенням обчислювального навантаження. Довжина фрейму 25 мс і крок 10 мс відповідають загальноприйнятим практикам обробки аудіосигналів і дозволяють адекватно відображати як стаціонарні, так і перехідні ділянки сигналу. Формування вхідного тензора розмірності $128 \times 40 \times 1$ забезпечує сумісність із CNN-архітектурою та уніфікує вхідні дані незалежно від тривалості початкового аудіо. Таким чином, таблиця 3.2 підтверджує коректність та обґрунтованість вибору параметрів експерименту.

Дані таблиці 3.3 дозволяють оцінити якість класифікації з використанням агрегованих метрик. Значення Accuracy = 0.81, macro-F1 = 0.81 та weighted-F1 = 0.81 є практично однаковими, що свідчить про збалансовану роботу моделі на всіх класах. Відсутність значної різниці між macro- та weighted-усередненням означає, що система не демонструє переваги для окремих жанрів за рахунок інших і не має критичних «провалів» по жодному з класів. Це є важливою характеристикою для практичного використання, оскільки модель поводиться стабільно незалежно від конкретного жанру вхідного аудіо.

Аналіз таблиці 3.4 дозволяє глибше зрозуміти характер помилок класифікації. Встановлено, що більшість помилкових рішень виникає між жанрами зі схожими часово-спектральними патернами, зокрема disco–pop, blues–rock та reggae–hip-hop. Це означає, що модель помиляється не випадково, а в межах логічно пояснюваних акустичних перекриттів між жанрами. Така поведінка є типовою для систем, що використовують MFCC-ознаки, і підтверджує коректність внутрішніх представлень, сформованих нейронною мережею.

Узагальнюючи результати, наведені в таблицях, можна зробити висновок, що розроблена система забезпечує стабільну та прогнозовану якість класифікації, демонструючи результати, близькі до відомих у науковій літературі для датасету GTZAN. Виявлені обмеження мають зрозумілу природу і не є критичними; навпаки, вони окреслюють напрями подальшого вдосконалення системи, зокрема шляхом аугментації даних, використання більш глибоких архітектур або залучення трансферного навчання.

Комплексний аналіз продемонстрував, що розроблена система забезпечує стабільну якість класифікації аудіоданих у межах задачі жанрового розпізнавання і досягає результатів, співставних з типовими показниками для датасету GTZAN. Поєднання MFCC і CNN у даній реалізації виступає практичним компромісом між обчислювальною складністю та точністю: MFCC забезпечують компактне й інформативне подання, а згорткова мережа ефективно виділяє локальні патерни у часо-коефіцієнтному просторі. Виявлені помилки мають зрозумілу акустичну природу (передусім для близьких жанрів), що є позитивною ознакою з позицій інженерного аналізу: такі помилки не виглядають випадковими, а відображають реальну складність розмежування стилів за короткими фрагментами.

Практична цінність системи посилюється наявністю графічного інтерфейсу та журналювання результатів, оскільки це переводить модель із формату «експерименту в ноутбуці» у формат прикладного програмного

інструмента. Водночас результати тесту зі співом птахів демонструють межі застосовності: для роботи з довільними звуками поза доменом музичних жанрів доцільно надалі додати механізм відмови (наприклад, порогування за максимальною ймовірністю або за ентропією розподілу), а також розглянути розширення навчального набору чи введення класу «невідомо». У сукупності наведені результати підтверджують, що реалізована система є коректною з точки зору інженерної логіки, відтворюваною за параметрами конвеєра та придатною як база для подальшого вдосконалення за рахунок аугментації даних, уточнення архітектури або застосування трансферного навчання.

У цьому розділі було реалізовано повноцінну комп’ютерну систему для класифікації електронних листів на спам та легітимні повідомлення. Запропонована система охоплює повний цикл обробки – від очищення тексту до надання результату користувачеві через зручний інтерфейс. Розроблено і протестовано кілька архітектур глибокого навчання: CNN, BERT та їхню гібридну комбінацію. Для кожної моделі здійснено навчання, оцінку продуктивності та порівняння результатів за ключовими метриками.

Результати експериментів показали, що гібридна модель CNN+BERT забезпечує найвищу точність класифікації (F1 $\approx$ 98.2 %), поєднуючи сильні сторони обох підходів: виявлення локальних шаблонів (CNN) і контекстуальний аналіз (BERT). Інтеграція з Flask дозволила реалізувати REST API та створити зручний веб-інтерфейс, що забезпечує взаємодію з користувачем у режимі реального часу. Структура проєкту збережена модульною, що спрощує супровід, масштабування та можливість подальших удосконалень.

Таким чином, поставлену задачу реалізації ефективної системи фільтрації спаму з використанням сучасних засобів машинного навчання повністю виконано. Розроблене рішення може бути використане як у дослідницьких цілях, так і для практичного впровадження в поштові сервіси.

# ВИСНОВКИ

В магістерській роботі розглянуті питання розробки програмної системи автоматичної класифікації адіоданих на основі згорткової нейромережі.

В першому розділі виконано огляд предметної області аудіокласифікації та розглянуто основні типи задач, що виникають під час автоматичного аналізу звукових сигналів, зокрема класифікацію музики, мовлення, акустичних сцен і подій. Проаналізовано класичні підходи на основі ручної інженерії ознак і статистичних класифікаторів, а також сучасні методи глибокого навчання, зокрема згорткові нейронні мережі, які забезпечують автоматичне виділення інформативних ознак і демонструють вищу точність у порівнянні з традиційними рішеннями.

На підставі огляду існуючих методів і систем сформульовано постановку задачі дослідження, що полягає у розробці прототипу програмної системи автоматичної класифікації аудіоданих на основі CNN з прикладною орієнтацією на класифікацію музичних фрагментів за жанрами. Визначено мету роботи, основні завдання та вимоги до архітектури системи, наборів даних, процесів попередньої обробки, навчання й оцінювання моделі.

У другому розділі виконано проєктування програмної системи класифікації аудіоданих та формалізовано повний конвеєр обробки від аудіофайлу до кінцевого рішення моделі. На підставі постановки задачі визначено й структуровано функціональні вимоги до ключових підсистем: введення та валідації аудіофайлів, попередньої обробки сигналу, виділення акустичних ознак, модулю нейромережевої класифікації та інтерфейсу користувача. Показано, що для забезпечення відтворюваності й стабільності результатів критично важливими є уніфікація параметрів аудіо (моно, ресемплінг, нормалізація), фреймінг із перекриттям та застосування віконних функцій перед спектральними перетвореннями.

Розроблено алгоритмічне забезпечення системи у вигляді формалізованих процедур: загального конвеєра класифікації, алгоритму попередньої обробки, формування MFCC-матриці для подачі в CNN та алгоритму інференсу з правилом вибору класу на основі максимуму ймовірності. Зафіксовано вимоги до узгодження розмірностей (обрізання/доповнення по часу, додавання канального виміру), нормалізації ознак і формування тензора фіксованого формату, що гарантує сумісність із архітектурою згорткової мережі та знижує ризик помилок, пов’язаних із “плаваючими” розмірами вхідних даних.

Запропоновано модульну архітектуру, побудовану на принципі розділення відповідальностей між компонентами: завантаження та валідація аудіо, препроцесинг, екстракція ознак, адаптація до тензорного формату, інференс CNN, прийняття рішення, представлення результатів і журналювання. Визначено потоки даних та їх формати на межах модулів, що створює підґрунтя для модульного тестування й подальшого розширення системи (заміна типу ознак, зміна кількості класів, підключення альтернативної моделі). UML-моделюванням формалізовано сценарії використання та послідовність взаємодії компонентів у базовому сценарії «Класифікувати аудіофайл», а також описано логіку обробки альтернативних і помилкових ситуацій.

Окремо обґрунтовано підхід до фіксації результатів класифікації через журналювання у структурованому форматі (наприклад, JSON), що забезпечує інженерну відтворюваність експериментів завдяки збереженню контексту запуску: параметрів аудіо, режиму ознак, версії моделі та показників упевненості. Таким чином, у розділі сформовано завершену проєктну основу для практичної реалізації системи: визначено вимоги, алгоритми, архітектуру, моделі взаємодії та формати даних, які будуть безпосередньо використані у наступному розділі під час реалізації й експериментального оцінювання ефективності..

У третьому розділі було реалізовано програмну систему автоматичної класифікації аудіоданих на основі зв’язки MFCC + CNN та виконано

експериментальне оцінювання її ефективності. Реалізація відповідає рішенням, спроєктованим у розділі 2, і побудована модульно: окремо виділено завантаження та валідацію аудіо, попередню обробку, обчислення MFCC, приведення ознак до тензора фіксованої розмірності, інференс CNN, інтерпретацію результатів і журналювання.

Для реалізації використано Python із бібліотеками NumPy/SciPy, TensorFlow/Keras, scikit-learn, а також Tkinter для створення графічного інтерфейсу. Система забезпечує повний конвеєр обробки «від файлу до рішення», повертає прогноз і топ-k альтернатив із показником впевненості, що підвищує практичну інформативність результатів.

Експерименти на тестовій підмножині GTZAN підтвердили стабільну якість: Accuracy ≈ 0.81 і узгоджені агреговані метрики (macro-F1 ≈ 0.81). Найкраще розпізнаються classical та metal, а основні помилки зосереджені між акустично близькими жанрами (передусім disco ↔ pop), що підтверджує не випадковий, а структурований характер помилок. Наявність GUI та журналювання переводить рішення з рівня експериментального прототипу до прикладного інструмента. Водночас тестування поза доменом показало доцільність подальшого розвитку — додавання механізму «відмови» та розширення даних/аугментації для підвищення стійкості до нетипових сигналів.

# ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1 Virtanen T., Plumbley M. D., Ellis D. P. W. (eds.). Computational Analysis of Sound Scenes and Events. Springer, 2018. DOI: 10.1007/978-3-319-63450-0. URL: https://link.springer.com/book/10.1007/978-3-319-63450-0 (дата звернення: 10.12.2025).

2 Stowell D., Giannoulis D., Benetos E., Lagrange M., Plumbley M. D. Detection and Classification of Acoustic Scenes and Events. IEEE Transactions on Multimedia. 2015. Vol. 17, No. 10. P. 1733–1746. DOI: 10.1109/TMM.2015.2428998. URL: https://www.researchgate.net/publication/276508708 (дата звернення: 10.12.2025).

3 Kinnunen T., Li H. An Overview of Text-Independent Speaker Recognition: From Features to Supervectors. Speech Communication. 2010. Vol. 52, No. 1. P. 12–40. DOI: 10.1016/j.specom.2009.08.009. URL: https://cs.joensuu.fi/sipu/pub/SRE-review-Kinnunen-and-Li.pdf (дата звернення: 10.12.2025).

4 MathWorks. Spectrogram (Short-Time Fourier Transform). URL: https://www.mathworks.com/help/signal/ref/spectrogram.html (дата звернення: 10.12.2025).

5 Stevens S. S., Volkmann J., Newman E. B. A Scale for the Measurement of the Psychological Magnitude Pitch. Journal of the Acoustical Society of America. 1937. Vol. 8, No. 3. P. 185–190. DOI: 10.1121/1.1915893. URL: https://pubs.aip.org/asa/jasa/article/8/3/185 (дата звернення: 10.12.2025).

6 Davis S., Mermelstein P. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing. 1980. Vol. 28, No. 4. P. 357–366. DOI: 10.1109/TASSP.1980.1163420. URL: https://courses.physics. illinois.edu/ ece417/fa2017/davis80.pdf (дата звернення: 10.12.2025).

7 McFee B., Raffel C., Liang D. та ін. librosa: Audio and Music Signal Analysis in Python. Proceedings of the 14th Python in Science Conference (SciPy 2015). URL: https://proceedings.scipy.org/articles/Majora-7b98e3ed-003.pdf (дата звернення: 10.12.2025).

8 librosa. Feature Extraction Documentation. URL: https://librosa.org/doc/0.11.0/feature.html (дата звернення: 10.12.2025).

9 Tzanetakis G., Cook P. Musical Genre Classification of Audio Signals. IEEE Transactions on Speech and Audio Processing. 2002. Vol. 10, No. 5. P. 293–302. DOI: 10.1109/TSA.2002.800560. URL: https://www.cs.cmu.edu/~gtzan/work/pubs/tsap02gtzan.pdf (дата звернення: 10.12.2025).

10 Kaggle. GTZAN Dataset — Music Genre Classification. URL: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification (дата звернення: 10.12.2025).

11 Salamon J., Jacoby C., Bello J. P. A Dataset and Taxonomy for Urban Sound Research. Proceedings of the 22nd ACM International Conference on Multimedia. 2014. URL: https://www.justinsalamon.com/uploads/4/3/9/4/4394963/salamon_urbansound_acmmm14.pdf (дата звернення: 10.12.2025).

12 Piczak K. J. ESC-50: Dataset for Environmental Sound Classification. URL: https://github.com/karolpiczak/ESC-50 (дата звернення: 10.12.2025).

13 Google Research. AudioSet: A Large-Scale Dataset of Audio Events. URL: https://research.google.com/audioset/download.html (дата звернення: 10.12.2025).

14 Hershey S., Chaudhuri S., Ellis D. P. W. та ін. CNN Architectures for Large-Scale Audio Classification. ICASSP 2017. URL: https://arxiv.org/abs/1609.09430 (дата звернення: 10.12.2025).

15 TensorFlow Models. VGGish: Audio Embedding Model. URL: https://github.com/tensorflow/models/blob/master/research/audioset/vggish/README.md (дата звернення: 10.12.2025).

16 TensorFlow Models. YAMNet: Pretrained Audio Event Classifier. URL: https://github.com/tensorflow/models/blob/master/research/audioset/yamnet/README.md (дата звернення: 10.12.2025).

17 Howard A. G., Zhu M., Chen B. та ін. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. URL: https://arxiv.org/abs/1704.04861 (дата звернення: 10.12.2025).

18 TensorFlow. Transfer Learning with YAMNet for Environmental Sound Classification. URL: https://www.tensorflow.org/tutorials/audio/transfer_learning_audio (дата звернення: 10.12.2025).

19 Wang A. L.-C. An Industrial-Strength Audio Search Algorithm. Proceedings of ISMIR 2003. URL: https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf (дата звернення: 10.12.2025).

20 Google. Google Assistant. URL: https://assistant.google.com/ (дата звернення: 10.12.2025).

21 Amazon. Alexa — Wake Word Technology. URL: https://www.amazon.com/b?ie=UTF8&node=23608571011 (дата звернення: 10.12.2025).

22 SoundHound AI. Official Website. URL: https://www.soundhound.com/ (дата звернення: 10.12.2025).

23 librosa. MFCC Feature Documentation. URL: https://librosa.org/doc/0.11.0/feature.html (дата звернення: 10.12.2025).

24 Barchiesi D., Giannoulis D., Stowell D., Plumbley M. D. Acoustic Scene Classification: A Tutorial and Survey. URL: https://arxiv.org/pdf/1411.3715 (дата звернення: 10.12.2025).

25 DCASE Community. Acoustic Scene Classification — Task Description. URL: https://dcase.community/challenge2021/task-acoustic-scene-classification (дата звернення: 10.12.2025).